NAME

netpbm - package of graphics manipulation programs and libraries

DESCRIPTION

Netpbm is a package of graphics programs and programming libraries.

There are over 220 separate programs in the package, most of which have "pbm", "pgm", "ppm", or "pnm" in their names. For example, **pnmscale** and **giftopnm**.

For example, you might use **pnmscale** to shrink an image by 10%. Or use **pnmcomp** to overlay one image on top of another. Or use **pbmtext** to create an image of text. Or reduce the number of colors in an image with **pnmquant**.

The Netpbm Formats

All of the programs work with a set of graphics formats called the "netpbm" formats. Specifically, these formats are **pbm**(5), **pgm**(5), **ppm**(5), and **pam**(5). The first three of these are sometimes known generically as **pnm**. Many of the Netpbm programs convert from a Netpbm format to another format or vice versa. This is so you can use the Netpbm programs to work on graphics of any format. It is also common to use a combination of Netpbm programs to convert from one non-Netpbm format to another non-Netpbm format. Netpbm has converters for over 80 graphics formats, and as a package Netpbm lets you do more graphics format conversions than any other computer graphics facility.

The Netpbm formats are all raster formats, i.e. they describe an image as a matrix of rows and columns of pixels. In the PBM format, the pixels are black and white. In the PGM format, pixels are shades of gray. In the PPM format, the pixels are in full color. The PAM format is more sophisticated. A replacement for all three of the other formats, it can represent matrices of general data including but not limited to black and white, grayscale, and color images.

Programs designed to work with PBM images have "pbm" in their names. Programs designed to work with PGM, PPM, and PAM images similarly have "pgm", "ppm", and "pam" in their names.

All Netpbm programs designed to read PGM images see PBM images as if they were PGM too. All Netpbm programs designed to read PPM images see PGM and PBM images as if they were PPM. See the section "Implied Format Conversion" below.

Programs that have "pnm" in their names read PBM, PGM, and PPM but unlike "ppm" programs, they distinguish between them and their function depends on the format. For example, **pnmtogif** creates a black and white GIF output image if its input is PBM or PGM, but a color GIF output image if its input is PPM. And **pnmscale** produces an output image of the same format as the input. A **ppmscale** program would read all three PNM input formats, but would see them all as PPM and would always generate PPM output.

If it seems wasteful to you to have three separate PNM formats, be aware that there is a historical reason for it. In the beginning, there were only PBMs. PGMs came later, and then PPMs. Much later came PAM, which realizes the possibility of having just one aggregate format.

The formats are described in the man pages **pbm**(5), **pgm**(5), **ppm**(5), and **pam**(5),

Implied Format Conversion

A program that uses the PGM library to read an image can read a PBM image as well as a PGM image. The program sees the PBM image as if it were the equivalent PGM image, with a maxval of 255.

A program that uses the PPM library to read an image can read a PGM image as well as a PPM image and a PBM image as well as a PGM image. The program sees the PBM or PGM image as if it were the equivalent PPM image, with a maxval of 255 in the PBM case and the same maxval as the PGM in the PGM case.

Netpbm and Transparency

In many graphics format, there's a means of indicating that certain parts of the image are wholly or partially transparent, meaning that if it were displayed "over" another image, the other image would show through there. Netpbm formats deliberately omit that capability, since their purpose is to be extremely simple.

In Netpbm, you handle transparency via a transparency mask in a separate (slightly redefined) PGM image. In this pseudo-PGM, what would normally be a pixel's intensity is instead it an opaqueness value. See **pgm**(5). **pnmcomp** is an example of a program that uses a PGM transparency mask.

The Netpbm Programs

The Netpbm programs are generally useful run by a person from a command shell, but are also designed to be used by programs. A common characteristic of Netpbm programs is that they are simple, fundamental building blocks. They are most powerful when stacked in pipelines. Netpbm programs do not use graphical user interfaces (in fact, none of them display graphics at all, except for a very simple Linux Svgalib displayer) and do not seek input from a user.

Each of these programs has its own man page.

Common Options

There are a few options that are present on all programs that are based on the Netpbm libraries, including virtually all Netpbm programs. These are not mentioned in the individual man pages for the programs.

-quiet Suppress all informational messages that would otherwise be issued to Standard Error. (To be precise, this only works to the extent that the program in question implements the Netpbm convention of issuing all informational messages via the **pm_message()** service of the Netpbm libraries).

-version
Instead of doing anything else, report the version of the **libpbm** library linked with the program (it may have been linked statically into the program, or dynamically linked at run time). Normally, the Netpbm programs and the libraries are installed at the same time, so this tells you the version of the program and all the other Netpbm libraries and files it uses as well.

Here is a directory of the Netpbm programs. You can also use **man -k** to search for a program that does what you want.

Converters

ppmtompeg
convert series of PPM frames to an MPEG movie

jpegtopnm
convert JFIF/JPEG/EXIF file to Netpbm format

pnmtojpeg
convert PPM to JPEG/JFIF/EXIF format

anytopnm
convert any graphics format to Netpbm format

bmptoppm
convert Windows or OS/2 Bitmap file to PPM

ppmtobmp
convert PPM to Windows or OS/2 Bitmap file

winicontoppm
convert Windows icon file to PPM

ppmtowinicon
convert PPM to Windows icon file

giftopnm
convert GIF to portable anymap

ppmtogif
convert PPM to GIF

pnmtopng
convert Netpbm format to Portable Network Graphics

pngtopnm
convert PNG (Portable Network Graphics) to Netpbm formats

palmtopnm
convert Palm pixmap to Netpbm formats

pnmtopalm
convert Netpbm formats to Palm pixmap

jbigtopbm
convert JBIG BIE (compressed bitmap) to PBM

pamtopnm
convert a PAM image to PBM, PGM, or PPM

pbmtojbig
convert PBM to JBIG BIE (compressed bitmap)

pnmtofiasco
convert Netpbm image to Fiasco (wfa) highly compressed format

fiascotopnm
convert Fiasco (wfa) highly compressed format to Netpbm image

hpcdtoppm
convert photo CD to PPM

pbmtonokia
convert PBM to Nokia Smart Messaging Format (SMF)

pbmtowbmp
convert PBM to WAP (Wireless App Protocol) Wireless Bitmap

wbmptopbm
convert WAP (Wireless App Protocol) Wireless Bitmap to PBM

neotoppm
convert Atari Neochrome (.neo) image to PPM

ppmtoneo
convert PPM image to Atari Neochrome (.neo)

pbmtomda
convert from PBM to Microdesign (for Amstrad PCWs)

mdatopbm
convert from Microdesign (for Amstrad PCWs) to PBM

atktopbm
convert Andrew Toolkit raster object to PBM

pbmtoatk
convert PBM to Andrew Toolkit raster object

brushtopbm
convert Xerox doodle brushes to PBM

cmuwmtopbm
convert CMU window manager format to PBM

g3topbm
convert Group 3 FAX to PBM

pbmtog3
convert PBM to Group 3 FAX

icontopbm
convert Sun icon to PBM

pbmtoicon
convert PBM to Sun icon

gemtopnm
convert GEM .img format to PBM or pixmap

macptopbm
convert MacPaint to PBM

pbmtomacp
convert PBM to MacPaint

mgrtopbm
convert MGR format to PBM

pbmtomgr
convert PBM to MGR format

pi3topbm
convert Atari Degas .pi3 to PBM

pbmtopi3
convert PBM to Atari Degas .pi3

xbmtopbm
convert X10 or X11 bitmap to PBM

pbmtoxbm
convert PBM to X11 bitmap

pbmtox10bm
convert PBM to X10 bitmap

ybmtopbm
convert Bennet Yee "face" file into PBM

pbmtoybm
convert PBM into Bennet Yee "face" file

pbmto10x
convert PBM to Gemini 10x printer graphics

pbmtoascii
convert PBM to ASCII graphic form

asciitopgm
convert ASCII character graphics to PGM

pbmtobbnbg
convert PBM to BBN BitGraph graphics

pbmtocmuwm
convert PBM to CMU window manager format

pbmtoepson
convert PBM to Epson printer graphics

pbmtogem
convert PBM into GEM .img file

pbmtogo
convert PBM to GraphOn graphics

pbmtolj
convert PBM to HP LaserJet black and white graphics

ppmtolj
convert PPM to HP LaserJet color graphics (PCL)

pjtoppm
convert HP PaintJet file to PPM

ppmtopj
convert PPM to HP PaintJet file

thinkjettopbm
convert HP Thinkjet printer stream to PBM

pbmtoplot
convert PBM into Unix plot(5) file

pbmtoptx
convert PBM to Printronix graphics

pbmtozinc
convert PBM to Zinc Interface Library icon

fitstopnm
convert FITS format to portable anymap

pnmtofits
convert Netpbm formats to FITS format

fstopgm
convert Usenix FaceSaver(tm) format to PGM

pgmtofs
convert PGM to Usenix FaceSaver(tm) format

hipstopgm
convert HIPS format to PGM

lispmtopgm
convert a Lisp Machine bitmap file into PGM format

pgmtolispm
convert PGM into Lisp Machine format

pnmtops
convert Netpbm formats to Postscript

pstopnm
convert Postscript to Netpbm formats

psidtopgm
convert PostScript "image" data to PGM

pbmtolps
convert PBM image to Postscript using lines

pbmtoepsi
convert a PBM image to encapsulated Postscript preview bitmap

pbmtopsg3
convert PBM images to Postscript using G3 fax compression.

rawtopgm
convert raw grayscale bytes to PGM

pgmtopbm
convert PGM to PBM

gouldtoppm
convert Gould scanner file to PPM

ilbmtoppm
convert IFF ILBM to PPM

ppmtoilbm
convert PPM to IFF ILBM

imgtoppm
convert Img-whatnot to PPM

mtvtoppm
convert MTV ray-tracer output to PPM

pcxtoppm
convert PC Paintbrush format to PPM

pgmtoppm
colorize a portable graymap into a PPM

pi1toppm
convert Atari Degas .pi1 to PPM

ppmtopi1
convert PPM to Atari Degas .pi1

picttoppm
convert Macintosh PICT to PPM

ppmtopict
convert PPM to Macintosh PICT

qrttoppm
convert QRT ray-tracer output to PPM

rawtoppm
convert raw RGB bytes to PPM

sldtoppm
convert an AutoCAD slide fi le into a PPM

spctoppm
convert Atari compressed Spectrum to PPM

sputoppm
convert Atari uncompressed Spectrum to PPM

tgatoppm
convert TrueVision Targa fi le to PPM

ppmtotga
convert PPM to TrueVision Targa fi le

ximtoppm
convert Xim to PPM

xpmtoppm
convert XPM format to PPM

ppmtoxpm
convert PPM to XPM format

yuvtoppm
convert Abekas YUV format to PPM

eyuvtoppm
convert Encoder/Berkeley YUV format to PPM

ppmtoeyuv
convert PPM to Encoder/Berkeley YUV format

ppmtoyuv
convert PPM to Abekas YUV format

ppmtoyuvsplit
convert PPM to 3 subsampled raw YUV files

yuvsplittoppm
merge 3 subsampled raw YUV files to one PPM

ppmtoacad
convert PPM to AutoCAD database or slide

ppmtoicr
convert PPM to NCSA ICR graphics

ppmtopcx
convert PPM to PC Paintbrush format

ppmtopgm
convert PPM to portable graymap

ppmtopuzz
convert PPM to X11 "puzzle" file

rasttopnm
convert Sun raster file to Netpbm formats

pnmtorast
convert Netpbm formats to Sun raster file

tifftopnm
convert TIFF file to portable anymap

pnmtotiff
convert Netpbm formats to TIFF RGB file

pnmtotiffcmyk
convert Netpbm formats to TIFF CMYK file

xwdtopnm
convert X10 or X11 window dump to Netpbm formats

pnmtoxwd
convert Netpbm formats to X11 window dump

pnmtoplainpnm
convert regular Netpbm format image into plain Netpbm format

pbmtopgm
convert PBM file to PGM by averaging areas

411toppm
convert 411 (Sony Mavica) to PPM

ppmtosixel
convert PPM to DEC sixel format

ppmtouil
convert PPM to Motif UIL icon file

sbigtopgm
convert Santa Barbara Instrument Group CCD file to PGM

vidtoppm
convert Parallax XVideo JPEG to sequence of PPM files

pnmtorle
convert PNM to Utah Raster Toolkit (urt/rle) file

rletopnm
convert Utah Raster Toolkit (urt/rle) file to PNM

ppmtoleaf
convert PPM to Interleaf

leaftoppm
convert Interleaf to PPM

bioradtopgm
convert Biorad confocal image to PGM

pbmtoln03
convert PGM image to Dec LN03+ Sixel image

pbmtopk
convert PBM image to packed format (PK) font

pktopbm
convert packed format (PK) font to PBM image

Image Generators

All of these generate Netpbm format output.

pbmmake
create a blank PBM image of a specifi ed size

ppmmake
create a PPM image of a specifi ed size and color

pgmramp
generate a grayscale ramp

ppmpat
create a pretty PPM image

ppmrainbow
create a spectrum-like image with colors fading together.

pgmnoise
create a PGM image of white noise

pbmtext
render text into a PBM image

pbmupc
create a Universal Product Code PBM image

ppmcie
generate a CIE color map PPM image

pbmpage
create a printer test pattern page in PBM format

ppmcolors
create a color map (PPM image) containing all possible colors of given maxval

Image Editors

All of these work on the Netpbm formats

ppmlabel
Add text to an image

pnmshadow
add a shadow to an image so it looks like it's floating

ppmbrighten
brighten or dim an image -- change saturation and value

ppmdim
dim an image - different way from ppmbrighten

pbmreduce
reduce a PBM N times, using Floyd-Steinberg

pgmnorm
normalize contrast in a PGM image

ppmnorm
normalize contrast in a PPM image

pbmpscale
enlarge a PBM image with edge smoothing

pnmscale
scale an image with high precision

pnmscalefixed
scale an image quickly with low precision

pnmenlarge
enlarge an image N times

ppmdither
ordered dither for color images

pnmcolormap
Choose the N best colors to represent an image; create a colormap

pnmremap
Replace colors in an image with those from a color map

ppmquant
quantize colors in a color image down to fewer colors

pnmquant
quantize colors/shades in a color or grayscale image down to fewer

ppmquantall
quantize colors on many files

ppmrelief
run a Laplacian Relief filter on a PPM

pnmarith
perform arithmetic on two images

pnmcat
concatenate images

pnmpad
add borders to an image

pnmcomp
create composite (overlay) of images

ppmmix
mix (overlay) two images.

pnmcrop
crop all like-colored borders off an image

pamcut
select a rectangular region from an image

pnmcut
obsolete version of **pamcut** (kept because it may have fewer bugs)

pamdice
slice an image into many horizontally and/or vertically

pamdeinterlace
remove every other row from an image

pamchannel
extract a single plane (channel, e.g. R, G, or B) from an image

pnmdepth
change the maxval in an image

pnmflip
perform one or more flip operations on an image

pamstretch
scale up an image by inserting interpolated pixels

pamstretch-gen
scale by non-integer values using pamstretch and pnmscale

pnminvert
invert an image

pnmgamma
perform gamma correction on an image

pnmhisteq
histogram equalize image to increase contrast

pnmmargin
add a margin to an image

pnmpaste
paste a rectangle into an image

pnmrotate
rotate an image

pnmshear
shear an image

pnmsmooth
smooth am image

pnmtile
replicate an image into a specifi ed size

pbmclean
remove lone pixels (snow) from a PBM image

pnmalias
antialias an image

ppmchange
change all of one color to another in PPM image

pnmnlfi lt
fi lter an image by replacing each pixel with a function of nearby pixels

ppmshift
shift lines of PPM image left or right a random amount

ppmspread
move pixels of PPM image a random amount

pnmconvol
general MxN convolution on an image

rgb3toppm
combine three portable graymaps into one PPM

ppmtorgb3
separate a PPM into three portable graymaps

pbmlife
apply Conway’s rules of Life to a PBM image

ppmdist
map colors to high contrast grayscales arbitrarily

ppmntsc
adjust colors so they are legal for NTSC or PAL television

Image Analyzers

These all work on the Netpbm formats as input.

pnmfi le
describe an image’s vital characteristics

pnmpsnr
measure difference between two images

pgmedge
edge-detect a PGM image

pgmenhance
edge-enhance a PGM image

pgmslice
print grayscale values for a row or column of a PGM image

pgmtexture
calculate textural features on a PGM image

pgmhist
print a histogram of the values in a PGM image

ppmhist
print a histogram of a PPM

pnmhistmap
draw a histogram of a PGM or PPM

ppmtomap
generate a map of all colors in an image

ppm3d generate a blue/green 3D glasses image from two images

Miscellaneous

ppmsvgalib
display a PPM image on a Linux virtual console using Svgalib

pbmmask
create a mask bitmap from a regular bitmap

ppmcolormask
create mask of areas of a certain color in an image

pnmsplit
split a multi-image Netpbm fi le into multiple 1-image fi les

pnmindex
build a visual index of a bunch of Netpbm images

pcdindex
build a visual index of a photo CD from PCD overview fi le

pnmmontage
build multiple Netpbm images into a single montage image

pgmbentley
Bentleyize a PGM image

pgmcrater	create cratered terrain by fractal forgery
pamoil	turn a PNM or PAM image into an oil painting
ppmforge	fractal forgeries of clouds, planets, and starry skies
pgmkernel	generate a convolution kernel
ppmtv	Make an image lined so it looks like an old TV
pbmto4425	Display PBM image on AT&T 4425 ASCII terminal with gfx chars

Uncatalogued As Yet

pnmtoddif

pnmtosgi

pnmtosir

ppmflash

ppmqvga

ppmtomitsu

ppmtopjxl

sgitopnm

sirtopnm

spottopgm

xvminitoppm

zeisstopnm

The Netpbm Libraries

The Netpbm programming libraries, **libpbm**(3), **libpgm**(3), **libppm**(3), and **libpnm**(3), make it easy to write programs that manipulate graphic images. Their main function is to read and write files in the Netpbm format, and because the Netpbm package contains converters for all the popular graphics formats, if your program reads and writes the Netpbm formats, you can use it with any formats.

But the libraries also contain some utility functions, such as character drawing and RGB/YCrCb conversion.

The libraries have the conventional C linkage. Virtually all programs in the Netpbm package are based on the Netpbm libraries.

Application Notes

As a collection of primitive tools, the power of Netpbm is multiplied by the power of all the other unix tools you can use with them. These notes remind you of some of the more useful ways to do this. Often, when people want to add high level functions to the Netpbm tools, they have overlooked some existing tool that, in combination with Netpbm, already does it.

Often, you need to apply some conversion or edit to a whole bunch of files.

As a rule, Netpbm programs take one input file and produce one output file, usually on Standard Output. This is for flexibility, since you so often have to pipeline many tools together.

Here is an example of a shell command to convert all your of PNG files (named *.png) to JPEG files named *.jpg:

```
for i in *.png; do pngtopnm $i | ppmtojpeg >‘basename $i .png‘.jpg; done
```

Or you might just generate a stream of individual shell commands, one per file, with awk or perl. Here’s how to brighten 30 YUV images that make up one second of a movie, keeping the images in the same files:

```
ls *.yuv .br | perl -ne ’chomp;
print yuvtoppm $_ | ppmbrighten -v 100 | ppmtoyuv >tmp$$.yuv; ,
mv tmp$$.yuv $_0
’ .br | sh
```

The tools **find** (with the **-exec** option) and **xargs** are also useful for simple manipulation of groups of files.

Some shells’ "process substitution" facility can help where a non-Netpbm program expects you to identify a disk file for input and you want it to use the result of a Netpbm manipulation. Say printcmyk takes the filename of a Tiff CMYK file as input and what you have is a PNG file **abc.png**. Try:

```
printcmyk <({ pngtopnm abc.png | pnmtotiffcmyk ; })
```

It works in the other direction too, if you have a program that makes you name its output file and you want the output to go through a Netpbm tool.

Other Graphics Software

Netpbm contains primitive building blocks. It certainly is not a complete graphics library.

The first thing you will want to make use of any of these tools is a viewer. (On GNU/Linux, you can use **ppmsvgalib** in a pinch, but it is pretty limiting). **zgv** is a good full service viewer to use on a GNU/Linux system with the SVGALIB graphics display driver library. You can find **zgv** at **ftp://ftp.ibiblio.org/pub/Linux/apps/graphics/viewers/svga .**

zgv even has a feature in it wherein you can visually crop an image and write an output file of the cropped image using **pnmcut**. See the **-s** option to **zgv**.

For the X inclined, there is also **xzgv**. See **ftp://metalab.unc.edu/pub/Linux/apps/graphics/viewers/X**.

xloadimage and its extension **xli** are also common ways to display a graphic image in X.

ImageMagick is like a visual version of Netpbm. Using the X/Window system on Unix, you can do basic editing of images and lots of format conversions. The package does include at least some non-visual tools. Convert, Mogrify, Montage, and Animate are popular programs from the **ImageMagick** package. **ImageMagick** runs on Unix, Windows, Windows NT, Macintosh, and VMS.

The Gimp is a visual image editor for Unix and X, in the same category as the more famous, less capable, and much more expensive Adobe Photoshop, etc. for Windows. See **http://www.gimp.org**.

The **file** program looks at a file and tells you what kind of file it is. It recognizes most of the graphics formats with which Netpbm deals, so it is pretty handy for graphics work. Netpbm’s **anytopnm** program depends on **file.** See **ftp://ftp.astron.com/pub/file**.

The Utah Raster Toolkit serves a lot of the same purpose as Netpbm, but without the emphasis on format conversions. This package is based on the RLE format, which you can convert to and from the Netpbm formats. **http://www.cs.utah.edu/research/projects/alpha1/urt.html** gives some information on the Utah Raster Toolkit, but does not tell where to get it.

There are some Netpbm-like graphics tools distributed by the Army High Performance Computing Research Center at **http://www.arc.umn.edu/gvl-software/media-tools.html**. These operate directly on non-Netpbm format images, so they aren't included in the Netpbm package. However, you can use them with any image format by using the Netpbm format converters.

Ivtools is a suite of free X Windows drawing editors for Postscript, Tex, and web graphics production, as well as an embeddable and extendable vector graphic shell. It uses the Netpbm facilities. See **http://www.ivtools.org**.

Ilib is a C subroutine library with functions for adding text to an image (as you might do at a higher level with **pbmtext**, **pnmcomp**, etc.). It works with Netpbm input and output. Find it at **http://www.radix.net/˜cknudsen/Ilib**. Netpbm also includes character drawing functions in the **libppm** library, but they do not have as fancy font capabilities (see **ppmlabel** for an example of use of the Netpbm character drawing functions).

GD is a library of graphics routines that is part of PHP. It has a subset of Netpbm's functions and has been found to resize images more slowly and with less quality.

pnm2ppa converts to HP's "Winprinter" format (for HP 710, 720, 820, 1000, etc). It is a superset of Netpbm's **pbmtoppa** and handles, notably, color. However, it is more of a printer driver than a Netpbm-style primitive graphics building block. See **http://sourceforge.net/project/?group_id=1322**.

The program **morph** morphs one image into another. It uses Targa format images, but you can use **tgatoppm** and **ppmtotga** to deal with that format. You have to use the graphical (X/Tk) Xmorph to create the mesh files that you must feed to **morph**. **morph** is part of the Xmorph package. See **http://www.colorado-research.com/˜gourlay/software/Graphics/Xmorph**.

To create an animated GIF, or extract a frame from one, use **gifsicle**. **gifsicle** converts between animated GIF and still GIF, and you can use **ppmtogif** and **giftopnm** to connect up to all the Netpbm utilities. See **http://www.lcdf.org/gifsicle**.

To convert an image of text to text (optical character recongition - OCR), use **gocr** (think of it as an inverse of **pbmtext**). See **http://altmark.nat.uni-magdeburg.de/˜jschulen/ocr/**.

http://schaik.com/pngsuite contains a PNG test suite -- a whole bunch of PNG images exploiting the various features of the PNG format.

Another version of **pnmtopng/pngtopnm** is at **http://www.schaik.com/png/pnmtopng.html**. The version in Netpbm was actually based on that package a long time ago, and you can expect to find better exploitation of the PNG format, especially recent enhancements, in that package. It may be a little less consistent with the Netpbm project and less exploitive of recent Netpbm format enhancements, though.

jpegtran Does some of the same transformations as Netpbm is famous for, but does them specifically on JPEG files and does them without loss of information. By contrast, if you were to use Netpbm, you would first decompress the JPEG image to Netpbm format, then transform the image, then compress it back to JPEG format. In that recompression, you lose a little image information because JPEG is a lossy compression. **jpegtran** comes with the Independent Jpeg Group's (http://www.ijg.org) JPEG library.

Some tools to deal with EXIF files (see also Netpbm's **jpegtopnm** and **pnmtojpeg**): To dump (interpret) an EXIF header: Exifdump ((http://topo.math.u-psud.fr/˜bousch/exifdump.py) or Jhead (http://www.sentex.net/˜mwandel/jhead).

A Python EXIF library and dumper: http://pyexif.sourceforge.net.

Latex2html converts Latex document source to HTML document source. Part of that involves graphics, and Latex2html uses Netpbm tools for some of that. But Latex2html through its history has had some rather esoteric codependencies with Netpbm. Older Latex2html doesn't work with current Netpbm. Latex2html-99.2beta8 works, though.

Other Graphics Formats

People never seem to tire of inventing new graphics formats, often completely redundant with preexisting ones. Netpbm cannot keep up with them. Here is a list of a few that we know Netpbm does *not* handle (yet).

CAL (originated by US Department Of Defense, favored by architects). http://www.landfield.com/faqs/graphics/fileformats-faq/part3/section-24.html

array formats dx, general, netcdf, CDF, hdf, cm

CGM+

Windows Meta File (.WMF). Libwmf converts from WMF to things like Latex, PDF, PNG. Some of these can be input to Netpbm.

Microsoft Word, RTF. Microsoft keeps a proprietary hold on these formats. Any software you see that can handle them is likely to cost money.

DXF (AutoCAD)

HISTORY

Netpbm has a long history, starting with Jef Poskanzer's **Pbmplus** package in 1988. The file *HISTORY* in the Netpbm source code contains a historical overview as well as a detailed history release by release.

AUTHOR

Netpbm is based on the **Pbmplus** package by Jef Poskanzer, first distributed in 1988 and maintained by him until 1991. But the package contains work by countless other authors, added since Jef's original work. In fact, the name is derived from the fact that the work was contributed by people all over the world via the Internet, when such collaboration was still novel enough to merit naming the package after it.

Bryan Henderson has been maintaining **Netpbm** since 1999. In addition to packaging work by others, Bryan has also written a significant amount of new material for the package.

NAME

411toppm - convert Sony Mavica .411 image to PPM

SYNOPSIS

411toppm [**-width** *width*] [**-height** *height*] [*411file*]

All options may be abbreviated to the shortest unique prefi x.

DESCRIPTION

Reads a .411 fi le, such as from a Sony Mavic camera, and converts it to a PPM image as output.

Output is to Standard Output.

The originator of this program and decipherer of the .411 format, Steve Allen <sla@alumni.caltech.edu>, has this to say about the utility of this program: "There's so little image in a 64x48 thumbnail (especially when you have the full size JPG fi le) that the only point in doing this was to answer the implicit challenge posed by the manual stating that only the camera can use these fi les."

OPTIONS

-width The width (number of columns) of the input image. Default is 64.

-height The height (number of rows) of the input image. Default is 48.

SEE ALSO

ppm(5)

NAME

anytopnm - convert an arbitrary type of image file to PBM, PGM, or PPM

SYNOPSIS

anytopnm [*file*]

DESCRIPTION

anytopnm converts the input image, which may be in any of dozens of graphics formats, to PBM, PGM, or PPM format, depending on that nature of the input image, and outputs it to Standard Output.

To determine the format of the input, **anytopnm** uses the **file** program (possibly assisted by the magic numbers file fragment included with Netpbm). If that fails (very few image formats have magic numbers), **anytopnm** looks at the filename extension. If that fails, **anytopnm** punts.

The type of the output file depends on the input image.

If **file** indicates that the input file is compressed (either via Unix compress, gzip, or bzip compression), **anytopnm** uncompresses it and proceeds as above with the uncompressed result.

If **file** indicates that the input file is encoded by uuencode or btoa, **anytopnm** decodes it and proceeds as above with the decoded result.

If *file* is **-** or not given, **anytopnm** takes its input from Standard Input.

SEE ALSO

pnmfile(1), **pnm**(5), **file**(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

asciitopgm - convert ASCII graphics into a portable graymap

SYNOPSIS

asciitopgm *[*-d divisor*] height width* [*asciifi le*]

DESCRIPTION

Reads ASCII data as input. Produces a portable graymap with pixel values which are an approximation of the "brightness" of the ASCII characters, assuming black-on-white printing. In other words, a capital M is very dark, a period is ver light, and a space is white. Input lines which are fewer than *width* characters are automatically padded with spaces.

The *divisor* argument is a floating-point number by which the output pixels are divided; the default value is 1.0. This can be used to adjust the brightness of the graymap: for example, if the image is too dim, reduce the divisor.

In keeping with (I believe) Fortran line-printer conventions, input lines beginning with a + (plus) character are assumed to "overstrike" the previous line, allowing a larger range of gray values.

This tool contradicts the message in the *pbmtoascii* manual: "Note that there is no asciitopbm tool - this transformation is one-way."

BUGS

The table of ASCII-to-grey values is subject to interpretation, and, of course, depends on the typeface intended for the input.

SEE ALSO

pbmtoascii(1), pgm(5)

AUTHOR

Wilson H. Bent. Jr. (whb@usc.edu)

NAME

atktopbm - convert Andrew Toolkit raster object to portable bitmap

SYNOPSIS

atktopbm [*atkfi le*]

DESCRIPTION

Reads an Andrew Toolkit raster object as input. Produces a portable bitmap as output.

SEE ALSO

pbmtoatk(1), pbm(5)

AUTHOR

Copyright (C) 1991 by Bill Janssen.

NAME

bioradtopgm - convert a Biorad confocal file into a portable graymap

SYNOPSIS

bioradtopgm [**-image#**] [*imagedata*]

DESCRIPTION

Reads a Biorad confocal file as input. Produces a portable graymap as output. If the resulting image is upside down, run it through **pnmflip -tb .**

OPTIONS

-image#

A Biorad image file may contain more than one image. With this flag, you can specify which image to extract (only one at a time). The first image in the file has number zero. If no image number is supplied, only information about the image size and the number of images in the input is printed out. No output is produced.

SEE ALSO

pgm(5), pnmflip(1)

AUTHORS

Copyright (C) 1993 by Oliver Trepte

NAME

bmptopnm − convert a BMP file into a PBM, PGM, or PNM image

SYNOPSIS

bmptopnm [*bmpfile*]

DESCRIPTION

Reads a Microsoft Windows or OS/2 BMP file as input. Produces a PBM, PGM, or PNM image as output. If the input is colormapped and contains only black and white, the output is PBM. If the input is colormapped and contains only black white and gray, the output is PGM. Otherwise, the output is PPM.

This program cannot convert BMP files with compressed (run length encoded) image data. It recognizes the compression and issues an error message.

This program cannot convert BMP files with 16 bits per pixel (only because the author did not have a complete specification for them). It recognizes the format and issues an error message.

SEE ALSO

ppmtobmp(1), **ppmtowinicon**(1), **ppm**(5)

AUTHOR

Copyright (C) 1992 by David W. Sanderson.

NAME

brushtopbm - convert a doodle brush fi le into a portable bitmap

SYNOPSIS

brushtopbm [*brushfi le*]

DESCRIPTION

Reads a Xerox doodle brush fi le as input. Produces a portable bitmap as output.

Note that there is currently no pbmtobrush tool.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

cmuwmtopbm - convert a CMU window manager bitmap into a portable bitmap

SYNOPSIS

cmuwmtopbm [*cmuwmfile*]

DESCRIPTION

Reads a CMU window manager bitmap as input. Produces a portable bitmap as output.

SEE ALSO

pbmtocmuwm(1), pbm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

eyuvtoppm - convert a Berkeley YUV file to a portable pixmap (ppm) file

SYNOPSIS

eyuvtoppm [**--width** *width*] [**--height** *height*] [*eyuvfile*]

DESCRIPTION

Reads a Berkeley Encoder YUV (not the same as Abekas YUV) file as input and produces a portable pixmap (ppm) file on the Standard Output.

With no filename argument takes input from Standard Input. Otherwise, *eyuvfile* is the file specification of the input file.

SEE ALSO

ppmtoeyuv(1), **yuvtoppm**(1), **ppm**(5)

NAME

fi ascotopnm − Convert compressed FIASCO image to PGM, or PPM

SYNOPSIS

fi ascotopnm [*option*]... [*fi lename*]...

DESCRIPTION

fi ascotopnm decompresses the named FIASCO fi les, or the Standard Input if no fi le is named, and writes the images as PGM, or PPM fi les, depending on whether the FIASCO image is black and white or color.

OPTIONS

All option names may be abbreviated; for example, --output may be written --outp or --ou. For all options an one letter short option is provided. Mandatory or optional arguments to long options are mandatory or optional for short options, too. Both short and long options are case sensitive.

−o[*name*], **−−output=**[*name*]

Write decompressed image to the fi le *name*.ppm (if PPM) or *name*.pgm (if PGM). If *name*=- then produce the image fi le on the standard output. The optional argument *name* can be omitted, then the input fi lename is used as basename with the suffi x .ppm or .pgm. In case of video streams, the frames are stored in the fi les *name*.**N**.ppm where **N** is the frame number (of the form 00..0 - 99..9); output on the standard output is not possible with video streams.

If *name* is a relative path and the environment variable **FIASCO_IMAGES** is a (colon-separated) list of directories, then the output fi le(s) are written to the fi rst (writable) directory of this list. Otherwise, the current directory is used to store the output fi le(s).

−z, **−−fast**

Decompress images in the 4:2:0 format; i.e., each chroma channel is decompressed to an image of halved width and height. Use this option on slow machines when the desired frame rate is not achieved; the output quality is only slightly decreased.

−d, **−−double**

Double the size of the X11 window both in width and height; no pixel interpolation is used, each pixel is just replaced by four identical pixels.

−p, **−−panel**

Show a panel with play, stop, pause, record and exit buttons to control the display of videos. When pressing the record button, all frames are decompressed and stored in memory. The other buttons work in the usual way.

−m *N*, **−−magnify=***N*

Set magnifi cation of the decompressed image. Positive values enlarge and negative values reduce the image width and height by a factor of 2ˆ|*N*|.

−s *N*, **−−smooth=***N*

Smooth decompressed image(s) along the partitioning borders by the given amount *N*. *N* is 1 (minimum) to 100 (maximum); default is 70. When *N*=0, then the smoothing amount specifi ed in the FIASCO fi le is used (defi ned by the FIASCO coder).

−F *N*, **−−fps=***N*

Set number of frames per second to *N*. When using this option, the frame rate specifi ed in the FIASCO fi le is overridden.

−v, **−−version**
Print **fiascotopnm** version number, then exit.

−f *name*, **−−config=***name*
Load parameter file *name* to initialize the options of **fiascotopnm**. See file **system.fiascorc** for an example of the syntax. Options of **fiascotopnm** are set by any of the following methods (in the specified order):

1) Global ressource file **/etc/system.fiascorc**

2) $HOME/**.fiascorc**

3) command line

4) --config=*name*

−h, **−−info**
Print brief help, then exit.

−H, **−−help**
Print detailed help, then exit.

EXAMPLES

fiascotopnm foo.wfa >foo.ppm
Decompress the FIASCO file "foo.wfa" and store it as "foo.ppm".

fiascotopnm -o foo1.wfa foo2.wfa
Decompress the FIASCO files "foo1.wfa" and "foo2.wfa" and write the frames to the image files "foo1.wfa.ppm" and "foo2.wfa.ppm".

fiascotopnm -oimage foo1.wfa
Decompress the FIASCO file "foo1.wfa" and write all 15 frames to the image files "image.00.ppm", ... , "image.14.ppm".

fiascotopnm --fast --magnify=-1 --double video.wfa >stream.ppm
Decompress the FIASCO file "video.wfa". The decompression speed is as fast as possible: the image is decompressed (in 4:2:0 format) at a quarter of its original size; then the image is enlarged again by pixel doubling.

FILES

/etc/system.fiascorc
The systemwide initialization file.

$HOME/**.fiascorc**
The personal initialization file.

ENVIRONMENT

FIASCO_IMAGES
Save path for image files. Default is "./".

FIASCO_DATA
Search path for FIASCO files. Default is "./".

SEE ALSO

pnmtofiasco(1), **pnm**(5)

Ullrich Hafner, Juergen Albert, Stefan Frank, and Michael Unger. **Weighted Finite Automata for Video Compression**, IEEE Journal on Selected Areas In Communications, January 1998
Ullrich Hafner. **Low Bit-Rate Image and Video Coding with Weighted Finite Automata**, Ph.D. thesis, Mensch & Buch Verlag, ISBN 3-89820-002-7, October 1999.

AUTHOR

Ullrich Hafner <hafner@bigfoot.de>

NAME

fitstopnm - convert a FITS file into a portable anymap

SYNOPSIS

fitstopnm [**-image** *N*] [**-noraw**] [**-scanmax**] [**-printmax**] [**-min** *f*] [**-max** *f*] [*FITSfile*]

DESCRIPTION

Reads a FITS file as input. Produces a portable pixmap if the FITS file consists of 3 image planes (NAXIS = 3 and NAXIS3 = 3), a portable graymap if the FITS file consists of 2 image planes (NAXIS = 2), or whenever the **−image** flag is specified. The results may need to be flipped top for bottom; if so, just pipe the output through **pnmflip -tb.**

OPTIONS

The **-image** option is for FITS files with three axes. The assumption is that the third axis is for multiple images, and this option lets you select which one you want.

Flags **-min** and **-max** can be used to override the min and max values as read from the FITS header or the image data if no DATAMIN and DATAMAX keywords are found. Flag **-scanmax** can be used to force the program to scan the data even when DATAMIN and DATAMAX are found in the header. If **-printmax** is specified, the program will just print the min and max values and quit. Flag **-noraw** can be used to force the program to produce an ASCII portable anymap.

The program will tell what kind of anymap is writing. All flags can be abbreviated to their shortest unique prefix.

REFERENCES

FITS stands for Flexible Image Transport System. A full description can be found in Astronomy & Astrophysics Supplement Series 44 (1981), page 363.

SEE ALSO

pnmtofits(1), pgm(5), pnmflip(1)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer, with modifications by Daniel Briggs (dbriggs@nrao.edu) and Alberto Accomazzi (alberto@cfa.harvard.edu).

NAME

fstopgm - convert a Usenix FaceSaver(tm) file into a portable graymap

SYNOPSIS

fstopgm [*fsfile*]

DESCRIPTION

Reads a Usenix FaceSaver(tm) file as input. Produces a portable graymap as output.

FaceSaver(tm) files sometimes have rectangular pixels. While *fstopgm* won't re-scale them into square pixels for you, it will give you the precise *pnmscale* command that will do the job. Because of this, reading a FaceSaver(tm) image is a two-step process. First you do:

```
  fstopgm > /dev/null
```

This will tell you whether you need to use *pnmscale.* Then use one of the following pipelines:

```
  fstopgm | pgmnorm
  fstopgm | pnmscale -whatever | pgmnorm
```

To go to PBM, you want something more like one of these:

```
  fstopgm | pnmenlarge 3 | pgmnorm | pgmtopbm
  fstopgm | pnmenlarge 3 | pnmscale <whatever> | pgmnorm | pgmtopbm
```

You want to enlarge when going to a bitmap because otherwise you lose information; but enlarging by more than 3 does not look good.

FaceSaver is a registered trademark of Metron Computerware Ltd. of Oakland, CA.

SEE ALSO

pgmtofs(1), pgm(5), pgmnorm(1), pnmenlarge(1), pnmscale(1), pgmtopbm(1)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

g3topbm - convert a Group 3 fax file into a portable bitmap

SYNOPSIS

g3topbm [**-kludge**] [**-reversebits**] [**-stretch**] [*g3file*]

DESCRIPTION

Reads a Group 3 fax file as input. Produces a portable bitmap as output.

OPTIONS

-kludge
Tells *g3topbm* to ignore the first few lines of the file; sometimes fax files have some junk at the beginning.

-reversebits
Tells *g3topbm* to interpret bits least-significant first, instead of the default most-significant first. Apparently some fax modems do it one way and others do it the other way. If you get a whole bunch of "bad code word" messages, try using this flag.

-stretch
Tells *g3topbm* to stretch the image vertically by duplicating each row. This is for the low-quality transmission mode.

All flags can be abbreviated to their shortest unique prefix.

REFERENCES

The standard for Group 3 fax is defined in CCITT Recommendation T.4.

BUGS

Probably.

SEE ALSO

pbmtog3(1), pbm(5)

AUTHOR

Copyright (C) 1989 by Paul Haeberli <paul@manray.sgi.com>.

NAME

giftopnm - convert a GIF file into a portable anymap

SYNOPSIS

giftopnm [**--alphaout=**{*alpha-filename*,**-**}] [**-verbose**] [**-comments**] [**-image** *N*] [*GIFfile*]

DESCRIPTION

This is a graphics format converter from the GIF format to the PNM (i.e. PBM, PGM, or PPM) format.

If the image contains only black and maximally bright white, the output is PBM. If the image contains more than those two colors, but only grays, the output is PGM. If the image contains other colors, the output is PPM.

If you have an animated GIF file, you can extract individual frames from it with **gifsicle** and then convert those using **giftopnm**.

A GIF image contains rectangular pixels. They all have the same aspect ratio, but may not be square (it's actually quite unusual for them not to be square, but it could happen). The pixels of a Netpbm image are always square. Because of the engineering complexity to do otherwise, **giftopnm** converts a GIF image to a Netpbm image pixel-for-pixel. This means if the GIF pixels are not square, the Netpbm output image has the wrong aspect ratio. In this case, **giftopnm** issues an informational message telling you to run **pnmscale** to correct the output.

OPTIONS

--alphaout=*alpha-filename*

giftopnm creates a PGM (portable graymap) file containing the alpha channel values in the input image. If the input image doesn't contain an alpha channel, the *alpha-filename* file contains all zero (transparent) alpha values. If you don't specify **--alphaout**, **giftopnm** does not generate an alpha file, and if the input image has an alpha channel, **giftopnm** simply discards it.

If you specify **-** as the filename, **giftopnm** writes the alpha output to Standard Output and discards the image.

See **pnmcomp**(1) for one way to use the alpha output file.

-verbose

Produce verbose output about the GIF file input.

-comments

Only output GIF89 comment fields.

-image *N*

Output the specified gif image from the input GIF archive (where *N* is '1', '2', '20'...). Normally there is only one image per file, so this option is not needed.

All flags can be abbreviated to their shortest unique prefix.

RESTRICTIONS

This does not correctly handle the Plain Text Extension of the GIF89 standard, since I did not have any example input files containing them.

SEE ALSO

ppmtogif(1), **ppmcolormask**(1), **pnmcomp**(1), **gifsicle**(1) <http://www.lcdf.org/gifsicle>, **ppm**(5).

AUTHOR

Copyright (c) 1993 by David Koblas (koblas@netcom.com)

LICENSE

If you use **giftopnm**, you are using a patent on the LZW compression method which is owned by Unisys, and in all probability you do not have a license from Unisys to do so. Unisys typically asks $5000 for a license for trivial use of the patent. Unisys has never enforced the patent against trivial users, and has made statements that it is much less concerned about people using the patent for decompression (which is what **giftopnm** does than for compression. The patent expires in 2003.

Rumor has it that IBM also owns a patent covering **giftopnm**.

A replacement for the GIF format that does not require any patents to use is the PNG format.

NAME

gouldtoppm - convert Gould scanner file into a portable pixmap

SYNOPSIS

gouldtoppm [*gouldfile*]

DESCRIPTION

Reads a file produced by the Gould scanner as input. Produces a portable pixmap as output.

SEE ALSO

ppm(5)

AUTHOR

Copyright(C) 1990 by Stephen Paul Lesniewski

NAME

hipstopgm - convert a HIPS file into a portable graymap

SYNOPSIS

hipstopgm [*hipsfile*]

DESCRIPTION

Reads a HIPS file as input. Produces a portable graymap as output.

If the HIPS file contains more than one frame in sequence, hipstopgm will concatenate all the frames vertically.

HIPS is a format developed at the Human Information Processing Laboratory, NYU.

SEE ALSO

pgm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

hpcdtoppm − convert a Photo-CD fi le into a portable bitmap fi le

SYNOPSIS

hpcdtoppm *infi le* [**−a**] [{**−C**|**−0**|**−Overview**|**−O**}] *fi le opt*] [**−c0**] [**−c-**] [**−c+**] [**−crop**] [**−d**] [**−dpi** *f*] [**−eps**] [**−epsd**] [**−epsg**] [**−fak** *scale*] [**−hori**] [**−i**] [**−l**] [**−m**] [**−n**] [**−pb** *pos*] [**−pgm**] [**−ph** *height*] [**−pl** *pos*] [**−pos**] [**−ppm**] [**−ps**] [**−psd**] [**−psg**] [**−pw** *width*] [**−r**] [**−rep**] [**−S** *long short*] [**−s**] [**−vert**] [**-x**] [**−ycc**] [**−1**|**−Base/16** |*−128x192*] [**−2**|**−Base/4** |*−256x384*] [**−3**|**−Base** |*−512x768*] [**−4**|**−4Base** |*−1024x1536*] [**−5**|**−16Base** |*−2048x3072*] [**−6**|**−64Base** |*−4096x6144*] [*outfi le*]

DESCRIPTION

This program accepts Photo-CD image or overview fi le data from the specifi ed input fi le, *infi le* (or, if the resolution is lower than 64Base and the fi le argument is specifi ed as −, from standard input), and writes either Portable Bitmap Format or POSTSCRIPT to the specifi ed output fi le (or to standard output if no fi le is specifi ed).

On a standard Photo-CD, image fi les appear in *photo_cd/images*, where they appear in fi les with names of the form img*nnnn.pcd*, where *nnnn* is a 4-digit-number. The overview fi le appears in *photo_cd/overview.pcd*.

Photo-CD images are stored using as many as 6 different resolutions:

Format	Resolution
------	----------
64Base	4096x6144 (ProPhotoCD only)
16Base	2048x3072
4Base	1024x1536
Base	512x768
Base/4	256x384
Base/16	128x192

The overview fi le employs Base/16 format.

OPTIONS

Invoking *hpcdtoppm* without arguments produces a list of default values. Note that you can supply only one size option.

−a Automatically determine image orientation (this option is experimental, and does not work for overview fi les).

{**−C** | **−0** | **−Overview** | **−O** } *fi le opt*
Extract all images from an overview fi le. The mandatory *fi le* argument is the name of a *ppm* fi le; output fi les are named *fi lennnn*, where *nnnn* is a 4-digit number. Overview images are extracted in their original Base/16 format. The value of *opt* determines the orientation of the contact sheet image; recognized values are:

n Do not rotate the image.

l Rotate the picture counter-clockwise (portrait mode).

r Rotate the picture clockwise (portrait mode).

−c0 Do not correct (brighten or darken) the image.

−c- Darken the image.

−c+ Brighten the image.

−crop Cut off the black frame which sometimes appears at the image borders.

−d Show only the decompressed difference rather than the complete image (applicable only to 4Base and 16Base images).

−dpi res
Set the printer resolution to *res* for dithered POSTSCRIPT images.

−eps Write a RGB Encapsulated POSTSCRIPT color image.

−epsd Write a Floyd-Steinberg dithered image in Encapsulated POSTSCRIPT.

−epsg Write a grayscale image in Encapsulated POSTSCRIPT.

−fak scale
Set the scaling factor for dithered POSTSCRIPT images to *scale*.

−hori Flip the image horizontally.

−i Send information from an image file header to standard error.

−l Rotate the picture counter-clockwise (portrait mode).

−m Write messages about the phases of decoding to standard error.

−n Do not rotate the image.

−pb pos
Set the bottom position of the POSTSCRIPT image to *pos*.

−pgm Write a *pgm* (grayscale) image.

−ph height
Set the height of the POSTSCRIPT image to *height*.

−pl pos Set the leftmost position of the POSTSCRIPT image to *pos*.

−pos Print the relative starting position of the data for the current resolution.

−ppm Write a *ppm* RGB (color) image.

−ps Write a RGB POSTSCRIPT color image.

−psd Write a Floyd-Steinberg dithered image in POSTSCRIPT.

−psg Write a POSTSCRIPT grayscale image.

−pw width
Set the width of the POSTSCRIPT image to *width*.

−r Rotate the picture clockwise (portrait mode).

−rep Try to jump over reading errors in the Huffman code.

−S long short
Cut out a subrectangle with boundaries defined by the values:

long For the longer side of the image.

short For the shorter side of the image.

where *long* and *short* take one of two forms:

a−b Cut from position *a* to position *b*.

a+b Starting at offset *a*, cut a length of *b*.

and where *a* and *b* are either integers representing pixel locations, or floating point values over the range [0.0 ... 1.0], representing the fraction of the length of a side.

−s Apply a simple sharpness operator to the luminosity channel.

−vert Flip the image vertically.

-x Overskip Mode (applicable to Base/16, Base/4, Base and 4Base). In Photo-CD images the luminosity channel is stored in full resolution, the two chromaticity channels are stored in half resolution only and have to be interpolated. In Overskip Mode, the chromaticity channels of the next higher resolution are taken instead of interpolating. To see the difference, generate one *ppm* with and one *ppm* without this flag. Use *pnmarith*(1L) to generate the difference image of these two images. Call *ppmhist*(1L) for this difference or show it with *xv*(1L) (push the **HistEq** button in the color editor).

−ycc Write the image in *ppm* YCC format.

−1|−Base/16|−128x192
Extract the Base/16 image.

−2 | −Base/4 | −256x384
Extract the Base/4 image.

−3 | −Base | −512x768
Extract the Base image.

−4 | −4Base | −1024x1536
Extract the 4Base image.

−5 | −16Base | −2048x3072
Extract the 16Base image.

−6 | −64Base | −4096x6144
Extract the 64Base image. This resolution can be extracted from ProPhotoCD images only. The path of the 64Base extension files is derived from the path to the image file. This means that it doesn't work on stdin an the directory structure must be the very same as on the ProPhotoCD.

POSTSCRIPT OUTPUT

For POSTSCRIPT output (options **−ps**, **−eps**, **−psg**, **−epsg**, **−psd**, **−epsg**) you can define both the resolution and placement of the image. Both size and position are specified in points (1/72 inch).

The position of the image (where the origin is assumed to be at the lower left corner of the page) is controlled by the **−pl** and **−pb** options (applicable at all resolutions).

The size of color and grayscale images is changed with the **−pw** and **−ph** options. Every image pixel is mapped onto one POSTSCRIPT pixel.

There are three modes of control for dithered POSTSCRIPT:

Image size
(**−pw** and **-ph**)

Printer resolution
(**−dpi**)

Scaling factor
(**−fak**)

These three factors are interdependent, hence no more then two can be specified simultaneously. Using **−dpi** and the **−pw**/**−ph** options together often yields pleasing results. Even using the default values for these options will produce results differing from those obtained without use of the options.

BUGS

The program ignores read protection.

The **−i** option is not working correctly.

Available information obout the Photo-CD format is vague; this program was developed by trial-and-error after staring at hex-dumps. Please send bugs reports and patches to the author.

SEE ALSO

pnmarith(1L), ppm(5L), ppmhist(1L), ppmquant(1L), ppmtopgm(1L), ppmtorgb3(1L), xv(1L)

VERSION

The name *hpcdtoppm* stands for "Hadmut's pcdtoppm," to make it distinguishable in the event that someone else is building a similar application and naming it *pcdtoppm*. This is version 0.6.

AUTHOR

Copyright (c) 1992, 1993, 1994 by Hadmut Danisch (danisch@ira.uka.de). This software is not public domain. Permission to use and distribute this software and its documentation for noncommercial use and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. This software may not be sold or used for profit-making activities.

Manual page extensively modified by R. P. C. Rodgers (rodgers@nlm.nih.gov).

NAME

icontopbm - convert a Sun icon into a portable bitmap

SYNOPSIS

icontopbm [*iconfile*]

DESCRIPTION

Reads a Sun icon as input. Produces a portable bitmap as output.

SEE ALSO

pbmtoicon(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME
icotoppm − convert bitmaps from ICO to ppm format

SYNOPSIS
icotoppm [*file*]

DESCRIPTION
icotoppm converts ICO bitmap files into PPM bitmaps. If no file is specified, the ICO file is read from standard input.

RESTRICTIONS
Multi icon files are not supported. Only RGB encoding is supported.

AUTHOR
Copyright (C) 1996 by Michael Haardt (michael@cantor.informatik.rwth-aachen.de).

SEE ALSO
ppm(5)

NAME

ilbmtoppm - convert an ILBM file into a portable pixmap

SYNOPSIS

ilbmtoppm [**-verbose**] [**-ignore**<chunkID>**]** [**-isham**|**-isehb**] [**-adjustcolors**] [*ILBMfile*]

DESCRIPTION

Reads an IFF ILBM file as input. Produces a portable pixmap as output. Supported ILBM types are:

Normal ILBMs with 1-16 planes.

Amiga Extra_Halfbrite (EHB)

Amiga HAM with 3-16 planes.

24 bit.

Multiplatte (normal or HAM) pictures.

Color map (BMHD + CMAP chunk only, nPlanes = 0).

Unofficial direct color.
  1-16 planes for each color component.

Chunks used:
  BMHD, CMAP, CAMG (only HAM & EHB flags used), PCHG, BODY unofficial DCOL chunk to identify direct color ILBM

Chunks ignored:
  GRAB, DEST, SPRT, CRNG, CCRT, CLUT, DPPV, DRNG, EPSF

Other chunks (ignored but displayed in verbose mode):
  NAME, AUTH, (c), ANNO, DPI

Unknown chunks are skipped.

OPTIONS

-verbose
  Give some information about the ILBM file.

-ignore <chunkID>
  Skip a chunk. <chunkID> is the 4-letter IFF chunk identifier of the chunk to be skipped.

-isham | -isehb
  Treat the input file as a HAM or Extra_Halfbrite picture, even if these flags or not set in the CAMG chunk (or if there is no CAMG chunk).

-adjustcolors
  If all colors in the CMAP have a value of less then 16, ilbmtoppm assumes a 4-bit colormap and gives a warning. With this option the colormap is scaled to 8 bits.

BUGS

The multipalette PCHG BigLineChanges and Huffman decompression code is untested.

REFERENCES

Amiga ROM Kernel Reference Manual - Devices (3rd Ed.)
Addison Wesley, ISBN 0-201-56775-X

SEE ALSO

ppm(5), ppmtoilbm(1)

AUTHORS

Copyright (C) 1989 by Jef Poskanzer.
Modified October 1993 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)

NAME

imgtoppm - convert an Img-whatnot fi le into a portable pixmap

SYNOPSIS

imgtoppm [*imgfi le*]

DESCRIPTION

Reads an Img-whatnot fi le as input. Produces a portable pixmap as output. The Img-whatnot toolkit is available for FTP on venera.isi.edu, along with numerous images in this format.

SEE ALSO

ppm(5)

AUTHOR

Based on a simple conversion program posted to comp.graphics by Ed Falk.

Copyright (C) 1989 by Jef Poskanzer.

NAME

jbigtopnm − JBIG to PNM image file converter

SYNOPSIS

jbigtopnm [*options*] [*input-file* | − [*output-file*]]

DESCRIPTION

Reads in a *JBIG* bi-level image entity (BIE) from a file or standard input, decompresses it, and outputs a PBM or PGM file. If the input has one plane, or you choose just one plane of it, the output is PBM. Otherwise, the output is PGM.

JBIG is a highly effective lossless compression algorithm for bi-level images (one bit per pixel), which is particularly suitable for scanned document pages.

A *JBIG* encoded image can be stored in several resolutions in one or several BIEs. All resolution layers except the lowest one are stored efficiently as differences to the next lower resolution layer. Options **-x** and **-y** can be used to stop the decompression at a specified maximal output image size. The input file can consist of several concatenated BIEs which contain different increasing resolution layers of the same image.

OPTIONS

−	A single hyphen instead of an input file name will cause **jbigtopnm** to read the data from standard input instead from a file.
−x *number*	Decode only up to the largest resolution layer which is still not more than *number* pixels wide. If no such resolution layer exists, then use the smallest one available.
−y *number*	Decode only up to the largest resolution layer which is still not more than *number* pixels high. If no such resolution layer exists, then use the smallest one available. Options **−x** and **−y** can also be used together in which case the largest layer that satisfies both limits will be selected.
−b	Use binary values instead of Gray code words in order to decode pixel values from multiple bitplanes. This option has only an effect if the input has more than one bitplane and you don't select just one of those bitplanes. Note that the decoder has to be used in the same mode as the encoder and cannot determine from the BIE, whether Gray or binary code words were used by the encoder.
−d	Diagnose a BIE. With this option, **jbigtopnm** will only print a summary of the header information found in the input file and then exit.
−p *number*	If the input contains multiple bitplanes, then extract only the specified single plane as a PBM file. The first plane has number 0.

STANDARDS

This program implements the *JBIG* image coding algorithm as specified in ISO/IEC 11544:1993 and ITU-T T.82(1993).

AUTHOR

The **jbigtopnm** is based on the *JBIG* library by Markus Kuhn, part of his **JBIG-KIT** package. The **jbgtopbm** program is part of the *JBIG-KIT* package. The most recent version of that library and tools set is freely available on the Internet from anonymous ftp server ftp.informatik.uni-erlangen.de in directory pub/doc/ISO/JBIG/.

jbigtopnm is part of the Netpbm package of graphics tools.

SEE ALSO

pbm(5),**pgm**(5),**pbmtojbg**(1)

LICENSE

If you use **jbigtopnm**, you are using various patents, particularly on its arithmetic encoding method, and in all probability you do not have a license from the patent owners to do so.

NAME

jpegtopnm − convert JPEG/JFIF file to portable pixmap or graymap

SYNOPSIS

jpegtopnm [**-dct** {**int**|**fast**|**float**}] [**-nosmooth**] [**-maxmemory** *N*] [{**-adobe**|**-notadobe**}] [**-comments**] [**-dumpexif**] [**-exif=***filespec*] [**-verbose**] [**-tracelevel** *N*] [*filename*]

All options may be abbreviated to their shortest unique prefix.

DESCRIPTION

jpegtopnm converts the named JFIF file, or the standard input if no file is named to a PPM or PGM image file on the standard output. If the JFIF file is of the grayscale variety, **jpegtopnm** generates a PGM (Portable Graymap) file. Otherwise, it generates a PPM (Portable Pixmap) file.

jpegtopnm uses the Independent JPEG Group's JPEG library to interpret the input file. See **http://www.ijg.org** for information on the library.

"JFIF" is the correct name for the image format commonly known as "JPEG." Strictly speaking, JPEG is a method of compression. The image format using JPEG compression that is by far the most common is JFIF. There is also a subformat of TIFF that uses JPEG compression.

EXIF is an image format that is a subformat of JFIF (to wit, a JFIF file that contains an EXIF header as an APP1 marker). **jpegtopnm** handles EXIF.

JFIF files can have either 8 bits per sample or 12 bits per sample. The 8 bit variety is by far the most common. There are two versions of the IJG JPEG library. One reads only 8 bit files and the other reads only 12 bit files. You must link the appropriate one of these libraries with **jpegtopnm**. Ordinarily, this means the library is in your shared library search path when you run **jpegtopnm**.

jpegtopnm generates output with either one byte or two bytes per sample depending on whether the JFIF input has either 8 bits or 12 bits per sample. You can use **pnmdepth** to reduce a two-byte-per-sample file to a one-byte-per-sample file if you need to.

If the JFIF file uses the CMYK or YCCK color space, the input does not actually contain enough information to know what color each pixel is. To know what color a pixel is, one would have to know the properties of the inks to which the color space refers. **jpegtopnm** interprets the colors using the common transformation which assumes all the inks are simply subtractive and linear.

OPTIONS

The options are only for advanced users:

−dct int
Use integer DCT method (default).

−dct fast
Use fast integer DCT (less accurate).

−dct float
Use floating-point DCT method. The float method is very slightly more accurate than the int method, but is much slower unless your machine has very fast floating-point hardware. Also note that results of the floating-point method may vary slightly across machines, while the integer methods should give the same results everywhere. The fast integer method is much less accurate than the other two.

−nosmooth
Use a faster, lower-quality upsampling routine.

−maxmemory *N*
Set limit on the amount of memory **jpegtopnm** uses in processing large images. Value is in thousands of bytes, or millions of bytes if "M" is suffixed to the number. For example, **−maxmemory 4m** selects 4000000 bytes. If **jpegtopnm** needs more space, it uses temporary

files.

−adobe

−notadobe
There are two variations on the CMYK (and likewise YCCK) color space that may be used in the JFIF input. In the normal one, a zero value for a color components indicates absence of ink. In the other, a zero value means the maximum ink coverage. The latter is used by Adobe Photoshop when it creates a bare JFIF output file (but not when it creates JFIF output as part of Encapsulated Postscript output).

These options tell **jpegtopnm** which version of the CMYK or YCCK color space the image uses. If you specify neither, **jpegtopnm** tries to figure it out on its own. In the present version, it doesn't try very hard at all: It just assumes the Photoshop version, since Photoshop and its emulators seem to be the main source of CMYK and YCCK images. But with experience of use, future versions might be more sophisticated.

If the JFIF image does not indicate that it is CMYK or YCCK, these options have no effect.

If you don't use the right one of these options, the symptom is output that looks like a negative.

−dumpexif
Print the interpreted contents of any Exif header in the input file to the Standard Error file. Similar to the program **jhead** (not part of the Netpbm package).

−exif=*filespec*
Extract the contents of the EXIF header from the input image and write it to the file *filespec*. *filespec* = **-** means write it to Standard Output. In this case, **jpegtopnm** does not output the converted image at all.

jpegtopnm writes the contents of the EXIF header byte-for-byte, starting with the two byte length field (which length includes those two bytes).

You can use this file as input to **ppmtojpeg** to insert an identical EXIF header into a new JFIF image.

If there is no EXIF header, **jpegtopnm** writes two bytes of binary zero and nothing else.

An EXIF header takes the form of a JFIF APP1 marker. Only the first such marker within the JFIF header counts.

−comments
Print any comments in the input file to the Standard Error file.

−verbose
Print details about the conversion to the Standard Error file.

−tracelevel *n*
Turn on the JPEG library's trace messages to the Standard Error file. A higher value of *n* gets more trace information. **−verbose** implies a trace level of at least 1.

EXAMPLES

This example converts the color JFIF file foo.jpg to a PPM file named foo.ppm:

```
jpegtopnm foo.jpg >foo.ppm
```

HINTS

You can use **ppmquant** to color quantize the result, i.e. to reduce the number of distinct colors in the image. In fact, you may have to if you want to convert the PPM file to certain other formats. **ppmdither** Does a more sophisticated quantization.

Use **pnmscale** to change the dimensions of the resulting image.

Use **ppmtopgm** to convert a color JFIF file to a grayscale PGM file.

You can easily use these converters together. E.g.:

```
jpegtopnm foo.jpg | ppmtopgm | pnmscale .25
>foo.pgm
```

−dct fast and/or **−nosmooth** gain speed at a small sacrifice in quality.

If you are fortunate enough to have very fast floating point hardware, **−dct float** may be even faster than **−dct fast**. But on most machines **−dct float** is slower than **−dct int**; in this case it is not worth using, because its theoretical accuracy advantage is too small to be significant in practice.

Another program, **djpeg**, is similar. **djpeg** is maintained by the Independent JPEG Group and packaged with the JPEG library which **jpegtopnm** uses for all its JPEG work. Because of that, you may expect it to exploit more current JPEG features. Also, since you have to have the library to run **jpegtopnm**, but not vice versa, **cjpeg** may be more commonly available.

On the other hand, **djpeg** does not use the NetPBM libraries to generate its output, as all the NetPBM tools such as **jpegtopnm** do. This means it is less likely to be consistent with all the other programs that deal with the NetPBM formats. Also, the command syntax of **jpegtopnm** is consistent with that of the other Netpbm tools, unlike **djpeg**.

ENVIRONMENT

JPEGMEM

If this environment variable is set, its value is the default memory limit. The value is specified as described for the **−maxmemory** option. An explicit **−maxmemory** option overrides any **JPEGMEM**.

SEE ALSO

ppm(5), **pgm**(5), **ppmtojpeg**(1), **ppmquant**(1), **pnmscale**(1), **ppmtopgm**(1), **ppmdither**(1), **pnmdepth**(1),
djpeg(1), **cjpeg**(1), **jpegtran**(1), **rdjpgcom**(1), **wrjpgcom**(1), **jhead**(1)
Wallace, Gregory K. "The JPEG Still Picture Compression Standard", Communications of the ACM, April 1991 (vol. 34, no. 4), pp. 30-44.

LIMITATIONS

Arithmetic coding is not supported for legal reasons.

The program could be much faster.

AUTHOR

jpegtopnm and this man page were derived in large part from **djpeg**, by the Independent JPEG Group. The program is otherwise by Bryan Henderson on March 19, 2000.

NAME

leaftoppm - convert Interleaf image format to PPM image

SYNOPSIS

leaftoppm [*leaffi le*]

DESCRIPTION

Reads a portable pixmap (PPM fi le) as input. Generates an Interleaf image fi le as output.

Interleaf is a now-defunct (actually purchased ca. 2000 by BroadVision) technical publishing software company.

SEE ALSO

ppm(5)

AUTHOR

The program is copyright (C) 1994 by Bill O'Donnell.

NAME

lispmtopgm - convert a Lisp Machine bitmap file into pgm format

SYNOPSIS

lispmtopgm [*lispmfile*]

DESCRIPTION

Reads a Lisp Machine bitmap as input. Produces a portable graymap as output.

This is the file format written by the tv:write-bit-array-file function on TI Explorer and Symbolics lisp machines.

Multi-plane bitmaps on lisp machines are color; but the lispm image file format does not include a color map, so we must treat it as a graymap instead. This is unfortunate.

SEE ALSO

pgmtolispm(1), pgm(5)

BUGS

The Lispm bitmap file format is a bit quirky; Usually the image in the file has its width rounded up to the next higher multiple of 32, but not always. If the width is not a multiple of 32, we don't deal with it properly, but because of the Lispm microcode, such arrays are probably not image data anyway.

Also, the lispm code for saving bitmaps has a bug, in that if you are writing a bitmap which is not mod32 across, the file may be up to 7 bits too short! They round down instead of up, and we don't handle this bug gracefully.

No color.

AUTHOR

Copyright (C) 1991 by Jamie Zawinski and Jef Poskanzer.

NAME

macptopbm - convert a MacPaint file into a portable bitmap

SYNOPSIS

macptopbm [**-extraskip** *N*] [*macpfile*]

DESCRIPTION

Reads a MacPaint file as input. Produces a portable bitmap as output.

OPTIONS

-extraskip

This flag is to get around a problem with some methods of transferring files from the Mac world to the Unix world. Most of these methods leave the Mac files alone, but a few of them add the "finderinfo" data onto the front of the Unix file. This means an extra 128 bytes to skip over when reading the file. The symptom to watch for is that the resulting PBM file looks shifted to one side. If you get this, try **-extraskip** 128, and if that still doesn't look right try another value.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

picttoppm(1), pbmtomacp(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer. The MacPaint-reading code is copyright (c) 1987 by Patrick J. Naughton (naughton@wind.sun.com).

NAME

mdatopbm - convert a Microdesign .mda or .mdp file into a portable bitmap

SYNOPSIS

mdatopbm [**-a**][**-d**][**-i**][**--**] *[mdafile]*

DESCRIPTION

Reads a MicroDesign file as input. Reads from stdin if input file is omitted. Produces a portable bitmap as output.

OPTIONS

-a Output the PBM in ASCII rather than binary.

-d Double the height of the output file, to compensate for the aspect ratio used in MicroDesign files.

-i Invert the colours used.

-- End of options (use this if the filename starts with "-")

SEE ALSO

pbmtomda(1), pbm(5)

AUTHOR

Copyright (C) 1999 John Elliott <jce@seasip.demon.co.uk>.

NAME

mgrtopbm - convert a MGR bitmap into a portable bitmap

SYNOPSIS

mgrtopbm [*mgrfi le*]

DESCRIPTION

Reads a MGR bitmap as input. Produces a portable bitmap as output.

SEE ALSO

pbmtomgr(1), pbm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

mtvtoppm - convert output from the MTV or PRT ray tracers into a portable pixmap

SYNOPSIS

mtvtoppm [*mtvfi le*]

DESCRIPTION

Reads an input fi le from Mark VanDeWettering's MTV ray tracer. Produces a portable pixmap as output.

The PRT raytracer also produces this format.

SEE ALSO

ppm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

neotoppm - convert an Atari Neochrome .neo into a PPM image

SYNOPSIS

neotoppm [*neofi le*]

DESCRIPTION

Reads an Atari Neochrome .neo fi le as input. Produces a portable pixmap as output.

SEE ALSO

ppmtoneo(1), **ppm**(5)

AUTHOR

Copyright (C) 2001 by Teemu Hukkanen <tjhukkan@iki.fi >, based on pi1toppm by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

netpbm - package of graphics manipulation programs and libraries

DESCRIPTION

Netpbm is a package of graphics programs and programming libraries.

There are over 220 separate programs in the package, most of which have "pbm", "pgm", "ppm", or "pnm" in their names. For example, **pnmscale** and **giftopnm**.

For example, you might use **pnmscale** to shrink an image by 10%. Or use **pnmcomp** to overlay one image on top of another. Or use **pbmtext** to create an image of text. Or reduce the number of colors in an image with **pnmquant**.

The Netpbm Formats

All of the programs work with a set of graphics formats called the "netpbm" formats. Specifically, these formats are **pbm**(5), **pgm**(5), **ppm**(5), and **pam**(5). The first three of these are sometimes known generically as **pnm**. Many of the Netpbm programs convert from a Netpbm format to another format or vice versa. This is so you can use the Netpbm programs to work on graphics of any format. It is also common to use a combination of Netpbm programs to convert from one non-Netpbm format to another non-Netpbm format. Netpbm has converters for over 80 graphics formats, and as a package Netpbm lets you do more graphics format conversions than any other computer graphics facility.

The Netpbm formats are all raster formats, i.e. they describe an image as a matrix of rows and columns of pixels. In the PBM format, the pixels are black and white. In the PGM format, pixels are shades of gray. In the PPM format, the pixels are in full color. The PAM format is more sophisticated. A replacement for all three of the other formats, it can represent matrices of general data including but not limited to black and white, grayscale, and color images.

Programs designed to work with PBM images have "pbm" in their names. Programs designed to work with PGM, PPM, and PAM images similarly have "pgm", "ppm", and "pam" in their names.

All Netpbm programs designed to read PGM images see PBM images as if they were PGM too. All Netpbm programs designed to read PPM images see PGM and PBM images as if they were PPM. See the section "Implied Format Conversion" below.

Programs that have "pnm" in their names read PBM, PGM, and PPM but unlike "ppm" programs, they distinguish between them and their function depends on the format. For example, **pnmtogif** creates a black and white GIF output image if its input is PBM or PGM, but a color GIF output image if its input is PPM. And **pnmscale** produces an output image of the same format as the input. A **ppmscale** program would read all three PNM input formats, but would see them all as PPM and would always generate PPM output.

If it seems wasteful to you to have three separate PNM formats, be aware that there is a historical reason for it. In the beginning, there were only PBMs. PGMs came later, and then PPMs. Much later came PAM, which realizes the possibility of having just one aggregate format.

The formats are described in the man pages **pbm**(5), **pgm**(5), **ppm**(5), and **pam**(5),

Implied Format Conversion

A program that uses the PGM library to read an image can read a PBM image as well as a PGM image. The program sees the PBM image as if it were the equivalent PGM image, with a maxval of 255.

A program that uses the PPM library to read an image can read a PGM image as well as a PPM image and a PBM image as well as a PGM image. The program sees the PBM or PGM image as if it were the equivalent PPM image, with a maxval of 255 in the PBM case and the same maxval as the PGM in the PGM case.

Netpbm and Transparency

In many graphics format, there's a means of indicating that certain parts of the image are wholly or partially transparent, meaning that if it were displayed "over" another image, the other image would show through there. Netpbm formats deliberately omit that capability, since their purpose is to be extremely simple.

In Netpbm, you handle transparency via a transparency mask in a separate (slightly redefined) PGM image. In this pseudo-PGM, what would normally be a pixel's intensity is instead it an opaqueness value. See **pgm**(5). **pnmcomp** is an example of a program that uses a PGM transparency mask.

The Netpbm Programs

The Netpbm programs are generally useful run by a person from a command shell, but are also designed to be used by programs. A common characteristic of Netpbm programs is that they are simple, fundamental building blocks. They are most powerful when stacked in pipelines. Netpbm programs do not use graphical user interfaces (in fact, none of them display graphics at all, except for a very simple Linux Svgalib displayer) and do not seek input from a user.

Each of these programs has its own man page.

Common Options

There are a few options that are present on all programs that are based on the Netpbm libraries, including virtually all Netpbm programs. These are not mentioned in the individual man pages for the programs.

-quiet Suppress all informational messages that would otherwise be issued to Standard Error. (To be precise, this only works to the extent that the program in question implements the Netpbm convention of issuing all informational messages via the **pm_message()** service of the Netpbm libraries).

-version
Instead of doing anything else, report the version of the **libpbm** library linked with the program (it may have been linked statically into the program, or dynamically linked at run time). Normally, the Netpbm programs and the libraries are installed at the same time, so this tells you the version of the program and all the other Netpbm libraries and files it uses as well.

Here is a directory of the Netpbm programs. You can also use **man -k** to search for a program that does what you want.

Converters

ppmtompeg
convert series of PPM frames to an MPEG movie

jpegtopnm
convert JFIF/JPEG/EXIF file to Netpbm format

pnmtojpeg
convert PPM to JPEG/JFIF/EXIF format

anytopnm
convert any graphics format to Netpbm format

bmptoppm
convert Windows or OS/2 Bitmap file to PPM

ppmtobmp
convert PPM to Windows or OS/2 Bitmap file

winicontoppm
convert Windows icon file to PPM

ppmtowinicon
convert PPM to Windows icon file

giftopnm
convert GIF to portable anymap

ppmtogif
convert PPM to GIF

pnmtopng
convert Netpbm format to Portable Network Graphics

pngtopnm
convert PNG (Portable Network Graphics) to Netpbm formats

palmtopnm
convert Palm pixmap to Netpbm formats

pnmtopalm
convert Netpbm formats to Palm pixmap

jbigtopbm
convert JBIG BIE (compressed bitmap) to PBM

pamtopnm
convert a PAM image to PBM, PGM, or PPM

pbmtojbig
convert PBM to JBIG BIE (compressed bitmap)

pnmtofiasco
convert Netpbm image to Fiasco (wfa) highly compressed format

fiascotopnm
convert Fiasco (wfa) highly compressed format to Netpbm image

hpcdtoppm
convert photo CD to PPM

pbmtonokia
convert PBM to Nokia Smart Messaging Format (SMF)

pbmtowbmp
convert PBM to WAP (Wireless App Protocol) Wireless Bitmap

wbmptopbm
convert WAP (Wireless App Protocol) Wireless Bitmap to PBM

neotoppm
convert Atari Neochrome (.neo) image to PPM

ppmtoneo
convert PPM image to Atari Neochrome (.neo)

pbmtomda
convert from PBM to Microdesign (for Amstrad PCWs)

mdatopbm
convert from Microdesign (for Amstrad PCWs) to PBM

atktopbm
convert Andrew Toolkit raster object to PBM

pbmtoatk
convert PBM to Andrew Toolkit raster object

brushtopbm
convert Xerox doodle brushes to PBM

cmuwmtopbm
convert CMU window manager format to PBM

g3topbm
convert Group 3 FAX to PBM

pbmtog3
convert PBM to Group 3 FAX

icontopbm
convert Sun icon to PBM

pbmtoicon
convert PBM to Sun icon

gemtopnm
convert GEM .img format to PBM or pixmap

macptopbm
convert MacPaint to PBM

pbmtomacp
convert PBM to MacPaint

mgrtopbm
convert MGR format to PBM

pbmtomgr
convert PBM to MGR format

pi3topbm
convert Atari Degas .pi3 to PBM

pbmtopi3
convert PBM to Atari Degas .pi3

xbmtopbm
convert X10 or X11 bitmap to PBM

pbmtoxbm
convert PBM to X11 bitmap

pbmtox10bm
convert PBM to X10 bitmap

ybmtopbm
convert Bennet Yee "face" file into PBM

pbmtoybm
convert PBM into Bennet Yee "face" file

pbmto10x
convert PBM to Gemini 10x printer graphics

pbmtoascii
convert PBM to ASCII graphic form

asciitopgm
convert ASCII character graphics to PGM

pbmtobbnbg
convert PBM to BBN BitGraph graphics

pbmtocmuwm
convert PBM to CMU window manager format

pbmtoepson
convert PBM to Epson printer graphics

pbmtogem
convert PBM into GEM .img file

pbmtogo
convert PBM to GraphOn graphics

pbmtolj
convert PBM to HP LaserJet black and white graphics

ppmtolj
convert PPM to HP LaserJet color graphics (PCL)

pjtoppm
convert HP PaintJet fi le to PPM

ppmtopj
convert PPM to HP PaintJet fi le

thinkjettopbm
convert HP Thinkjet printer stream to PBM

pbmtoplot
convert PBM into Unix plot(5) fi le

pbmtoptx
convert PBM to Printronix graphics

pbmtozinc
convert PBM to Zinc Interface Library icon

fi tstopnm
convert FITS format to portable anymap

pnmtofi ts
convert Netpbm formats to FITS format

fstopgm
convert Usenix FaceSaver(tm) format to PGM

pgmtofs
convert PGM to Usenix FaceSaver(tm) format

hipstopgm
convert HIPS format to PGM

lispmtopgm
convert a Lisp Machine bitmap fi le into PGM format

pgmtolispm
convert PGM into Lisp Machine format

pnmtops
convert Netpbm formats to Postscript

pstopnm
convert Postscript to Netpbm formats

psidtopgm
convert PostScript "image" data to PGM

pbmtolps
convert PBM image to Postscript using lines

pbmtoepsi
convert a PBM image to encapsulated Postscript preview bitmap

pbmtopsg3
convert PBM images to Postscript using G3 fax compression.

rawtopgm
convert raw grayscale bytes to PGM

pgmtopbm
convert PGM to PBM

gouldtoppm
convert Gould scanner fi le to PPM

ilbmtoppm
convert IFF ILBM to PPM

ppmtoilbm
convert PPM to IFF ILBM

imgtoppm
convert Img-whatnot to PPM

mtvtoppm
convert MTV ray-tracer output to PPM

pcxtoppm
convert PC Paintbrush format to PPM

pgmtoppm
colorize a portable graymap into a PPM

pi1toppm
convert Atari Degas .pi1 to PPM

ppmtopi1
convert PPM to Atari Degas .pi1

picttoppm
convert Macintosh PICT to PPM

ppmtopict
convert PPM to Macintosh PICT

qrttoppm
convert QRT ray-tracer output to PPM

rawtoppm
convert raw RGB bytes to PPM

sldtoppm
convert an AutoCAD slide fi le into a PPM

spctoppm
convert Atari compressed Spectrum to PPM

sputoppm
convert Atari uncompressed Spectrum to PPM

tgatoppm
convert TrueVision Targa fi le to PPM

ppmtotga
convert PPM to TrueVision Targa fi le

ximtoppm
convert Xim to PPM

xpmtoppm
convert XPM format to PPM

ppmtoxpm
convert PPM to XPM format

yuvtoppm
convert Abekas YUV format to PPM

eyuvtoppm
convert Encoder/Berkeley YUV format to PPM

ppmtoeyuv
convert PPM to Encoder/Berkeley YUV format

ppmtoyuv
convert PPM to Abekas YUV format

ppmtoyuvsplit
convert PPM to 3 subsampled raw YUV files

yuvsplittoppm
merge 3 subsampled raw YUV files to one PPM

ppmtoacad
convert PPM to AutoCAD database or slide

ppmtoicr
convert PPM to NCSA ICR graphics

ppmtopcx
convert PPM to PC Paintbrush format

ppmtopgm
convert PPM to portable graymap

ppmtopuzz
convert PPM to X11 "puzzle" file

rasttopnm
convert Sun raster file to Netpbm formats

pnmtorast
convert Netpbm formats to Sun raster file

tifftopnm
convert TIFF file to portable anymap

pnmtotiff
convert Netpbm formats to TIFF RGB file

pnmtotiffcmyk
convert Netpbm formats to TIFF CMYK file

xwdtopnm
convert X10 or X11 window dump to Netpbm formats

pnmtoxwd
convert Netpbm formats to X11 window dump

pnmtoplainpnm
convert regular Netpbm format image into plain Netpbm format

pbmtopgm
convert PBM file to PGM by averaging areas

411toppm
convert 411 (Sony Mavica) to PPM

ppmtosixel
convert PPM to DEC sixel format

ppmtouil
convert PPM to Motif UIL icon file

sbigtopgm
convert Santa Barbara Instrument Group CCD file to PGM

vidtoppm
convert Parallax XVideo JPEG to sequence of PPM files

pnmtorle
convert PNM to Utah Raster Toolkit (urt/rle) file

rletopnm
convert Utah Raster Toolkit (urt/rle) file to PNM

ppmtoleaf
convert PPM to Interleaf

leaftoppm
convert Interleaf to PPM

bioradtopgm
convert Biorad confocal image to PGM

pbmtoln03
convert PGM image to Dec LN03+ Sixel image

pbmtopk
convert PBM image to packed format (PK) font

pktopbm
convert packed format (PK) font to PBM image

Image Generators

All of these generate Netpbm format output.

pbmmake
create a blank PBM image of a specifi ed size

ppmmake
create a PPM image of a specifi ed size and color

pgmramp
generate a grayscale ramp

ppmpat
create a pretty PPM image

ppmrainbow
create a spectrum-like image with colors fading together.

pgmnoise
create a PGM image of white noise

pbmtext
render text into a PBM image

pbmupc
create a Universal Product Code PBM image

ppmcie
generate a CIE color map PPM image

pbmpage
create a printer test pattern page in PBM format

ppmcolors
create a color map (PPM image) containing all possible colors of given maxval

Image Editors

All of these work on the Netpbm formats

ppmlabel
Add text to an image

pnmshadow
add a shadow to an image so it looks like it's fbating

ppmbrighten
brighten or dim an image -- change saturation and value

ppmdim
dim an image - different way from ppmbrighten

pbmreduce
reduce a PBM N times, using Floyd-Steinberg

pgmnorm
normalize contrast in a PGM image

ppmnorm
normalize contrast in a PPM image

pbmpscale
enlarge a PBM image with edge smoothing

pnmscale
scale an image with high precision

pnmscalefixed
scale an image quickly with low precision

pnmenlarge
enlarge an image N times

ppmdither
ordered dither for color images

pnmcolormap
Choose the N best colors to represent an image; create a colormap

pnmremap
Replace colors in an image with those from a color map

ppmquant
quantize colors in a color image down to fewer colors

pnmquant
quantize colors/shades in a color or grayscale image down to fewer

ppmquantall
quantize colors on many files

ppmrelief
run a Laplacian Relief filter on a PPM

pnmarith
perform arithmetic on two images

pnmcat
concatenate images

pnmpad
add borders to an image

pnmcomp
create composite (overlay) of images

ppmmix
mix (overlay) two images.

pnmcrop
crop all like-colored borders off an image

pamcut
select a rectangular region from an image

pnmcut
obsolete version of **pamcut** (kept because it may have fewer bugs)

pamdice
slice an image into many horizontally and/or vertically

pamdeinterlace
remove every other row from an image

pamchannel
extract a single plane (channel, e.g. R, G, or B) from an image

pnmdepth
change the maxval in an image

pnmflip
perform one or more flip operations on an image

pamstretch
scale up an image by inserting interpolated pixels

pamstretch-gen
scale by non-integer values using pamstretch and pnmscale

pnminvert
invert an image

pnmgamma
perform gamma correction on an image

pnmhisteq
histogram equalize image to increase contrast

pnmmargin
add a margin to an image

pnmpaste
paste a rectangle into an image

pnmrotate
rotate an image

pnmshear
shear an image

pnmsmooth
smooth am image

pnmtile
replicate an image into a specifi ed size

pbmclean
remove lone pixels (snow) from a PBM image

pnmalias
antialias an image

ppmchange
change all of one color to another in PPM image

pnmnlfi lt
fi lter an image by replacing each pixel with a function of nearby pixels

ppmshift
shift lines of PPM image left or right a random amount

ppmspread
move pixels of PPM image a random amount

pnmconvol
general MxN convolution on an image

rgb3toppm
combine three portable graymaps into one PPM

ppmtorgb3
separate a PPM into three portable graymaps

pbmlife
apply Conway's rules of Life to a PBM image

ppmdist
map colors to high contrast grayscales arbitrarily

ppmntsc
adjust colors so they are legal for NTSC or PAL television

Image Analyzers

These all work on the Netpbm formats as input.

pnmfile
describe an image's vital characteristics

pnmpsnr
measure difference between two images

pgmedge
edge-detect a PGM image

pgmenhance
edge-enhance a PGM image

pgmslice
print grayscale values for a row or column of a PGM image

pgmtexture
calculate textural features on a PGM image

pgmhist
print a histogram of the values in a PGM image

ppmhist
print a histogram of a PPM

pnmhistmap
draw a histogram of a PGM or PPM

ppmtomap
generate a map of all colors in an image

ppm3d generate a blue/green 3D glasses image from two images

Miscellaneous

ppmsvgalib
display a PPM image on a Linux virtual console using Svgalib

pbmmask
create a mask bitmap from a regular bitmap

ppmcolormask
create mask of areas of a certain color in an image

pnmsplit
split a multi-image Netpbm file into multiple 1-image files

pnmindex
build a visual index of a bunch of Netpbm images

pcdindex
build a visual index of a photo CD from PCD overview file

pnmmontage
build multiple Netpbm images into a single montage image

pgmbentley
Bentleyize a PGM image

pgmcrater
create cratered terrain by fractal forgery

pamoil turn a PNM or PAM image into an oil painting

ppmforge
fractal forgeries of clouds, planets, and starry skies

pgmkernel
generate a convolution kernel

ppmtv Make an image lined so it looks like an old TV

pbmto4425
Display PBM image on AT&T 4425 ASCII terminal with gfx chars

Uncatalogued As Yet

pnmtoddif

pnmtosgi

pnmtosir

ppmflash

ppmqvga

ppmtomitsu

ppmtopjxl

sgitopnm

sirtopnm

spottopgm

xvminitoppm

zeisstopnm

The Netpbm Libraries

The Netpbm programming libraries, **libpbm**(3), **libpgm**(3), **libppm**(3), and **libpnm**(3), make it easy to write programs that manipulate graphic images. Their main function is to read and write files in the Netpbm format, and because the Netpbm package contains converters for all the popular graphics formats, if your program reads and writes the Netpbm formats, you can use it with any formats.

But the libraries also contain some utility functions, such as character drawing and RGB/YCrCb conversion.

The libraries have the conventional C linkage. Virtually all programs in the Netpbm package are based on the Netpbm libraries.

Application Notes

As a collection of primitive tools, the power of Netpbm is multiplied by the power of all the other unix tools you can use with them. These notes remind you of some of the more useful ways to do this. Often, when people want to add high level functions to the Netpbm tools, they have overlooked some existing tool that, in combination with Netpbm, already does it.

Often, you need to apply some conversion or edit to a whole bunch of files.

As a rule, Netpbm programs take one input file and produce one output file, usually on Standard Output. This is for flexibility, since you so often have to pipeline many tools together.

Here is an example of a shell command to convert all your of PNG files (named *.png) to JPEG files named *.jpg:

```
for i in *.png; do pngtopnm $i | ppmtojpeg >‘basename $i .png‘.jpg; done
```

Or you might just generate a stream of individual shell commands, one per file, with awk or perl. Here’s how to brighten 30 YUV images that make up one second of a movie, keeping the images in the same files:

```
ls *.yuv .br | perl -ne ’chomp;
print yuvtoppm $_ | ppmbrighten -v 100 | ppmtoyuv >tmp$$.yuv; ,
mv tmp$$.yuv $_0
’ .br | sh
```

The tools **find** (with the **-exec** option) and **xargs** are also useful for simple manipulation of groups of files.

Some shells’ "process substitution" facility can help where a non-Netpbm program expects you to identify a disk file for input and you want it to use the result of a Netpbm manipulation. Say printcmyk takes the filename of a Tiff CMYK file as input and what you have is a PNG file **abc.png**. Try:

```
printcmyk <({ pngtopnm abc.png | pnmtotiffcmyk ; })
```

It works in the other direction too, if you have a program that makes you name its output file and you want the output to go through a Netpbm tool.

Other Graphics Software

Netpbm contains primitive building blocks. It certainly is not a complete graphics library.

The first thing you will want to make use of any of these tools is a viewer. (On GNU/Linux, you can use **ppmsvgalib** in a pinch, but it is pretty limiting). **zgv** is a good full service viewer to use on a GNU/Linux system with the SVGALIB graphics display driver library. You can find **zgv** at **ftp://ftp.ibiblio.org/pub/Linux/apps/graphics/viewers/svga .**

zgv even has a feature in it wherein you can visually crop an image and write an output file of the cropped image using **pnmcut**. See the **-s** option to **zgv**.

For the X inclined, there is also **xzgv**. See **ftp://metalab.unc.edu/pub/Linux/apps/graphics/viewers/X**.

xloadimage and its extension **xli** are also common ways to display a graphic image in X.

ImageMagick is like a visual version of Netpbm. Using the X/Window system on Unix, you can do basic editing of images and lots of format conversions. The package does include at least some non-visual tools. Convert, Mogrify, Montage, and Animate are popular programs from the **ImageMagick** package. **ImageMagick** runs on Unix, Windows, Windows NT, Macintosh, and VMS.

The Gimp is a visual image editor for Unix and X, in the same category as the more famous, less capable, and much more expensive Adobe Photoshop, etc. for Windows. See **http://www.gimp.org**.

The **file** program looks at a file and tells you what kind of file it is. It recognizes most of the graphics formats with which Netpbm deals, so it is pretty handy for graphics work. Netpbm’s **anytopnm** program depends on **file.** See **ftp://ftp.astron.com/pub/file**.

The Utah Raster Toolkit serves a lot of the same purpose as Netpbm, but without the emphasis on format conversions. This package is based on the RLE format, which you can convert to and from the Netpbm formats. **http://www.cs.utah.edu/research/projects/alpha1/urt.html** gives some information on the Utah Raster Toolkit, but does not tell where to get it.

There are some Netpbm-like graphics tools distributed by the Army High Performance Computing Research Center at **http://www.arc.umn.edu/gvl-software/media-tools.html**. These operate directly on non-Netpbm format images, so they aren't included in the Netpbm package. However, you can use them with any image format by using the Netpbm format converters.

Ivtools is a suite of free X Windows drawing editors for Postscript, Tex, and web graphics production, as well as an embeddable and extendable vector graphic shell. It uses the Netpbm facilities. See **http://www.ivtools.org**.

Ilib is a C subroutine library with functions for adding text to an image (as you might do at a higher level with **pbmtext**, **pnmcomp**, etc.). It works with Netpbm input and output. Find it at **http://www.radix.net/˜ cknudsen/Ilib**. Netpbm also includes character drawing functions in the **libppm** library, but they do not have as fancy font capabilities (see **ppmlabel** for an example of use of the Netpbm character drawing functions).

GD is a library of graphics routines that is part of PHP. It has a subset of Netpbm's functions and has been found to resize images more slowly and with less quality.

pnm2ppa converts to HP's "Winprinter" format (for HP 710, 720, 820, 1000, etc). It is a superset of Netpbm's **pbmtoppa** and handles, notably, color. However, it is more of a printer driver than a Netpbm-style primitive graphics building block. See **http://sourceforge.net/project/?group_id=1322**.

The program **morph** morphs one image into another. It uses Targa format images, but you can use **tgatoppm** and **ppmtotga** to deal with that format. You have to use the graphical (X/Tk) Xmorph to create the mesh files that you must feed to **morph**. **morph** is part of the Xmorph package. See **http://www.colorado-research.com/˜ gourlay/software/Graphics/Xmorph**.

To create an animated GIF, or extract a frame from one, use **gifsicle**. **gifsicle** converts between animated GIF and still GIF, and you can use **ppmtogif** and **giftopnm** to connect up to all the Netpbm utilities. See **http://www.lcdf.org/gifsicle**.

To convert an image of text to text (optical character recongition - OCR), use **gocr** (think of it as an inverse of **pbmtext**). See **http://altmark.nat.uni-magdeburg.de/˜ jschulen/ocr/**.

http://schaik.com/pngsuite contains a PNG test suite -- a whole bunch of PNG images exploiting the various features of the PNG format.

Another version of **pnmtopng/pngtopnm** is at **http://www.schaik.com/png/pnmtopng.html**. The version in Netpbm was actually based on that package a long time ago, and you can expect to find better exploitation of the PNG format, especially recent enhancements, in that package. It may be a little less consistent with the Netpbm project and less exploitive of recent Netpbm format enhancements, though.

jpegtran Does some of the same transformations as Netpbm is famous for, but does them specifically on JPEG files and does them without loss of information. By contrast, if you were to use Netpbm, you would first decompress the JPEG image to Netpbm format, then transform the image, then compress it back to JPEG format. In that recompression, you lose a little image information because JPEG is a lossy compression. **jpegtran** comes with the Independent Jpeg Group's (http://www.ijg.org) JPEG library.

Some tools to deal with EXIF files (see also Netpbm's **jpegtopnm** and **pnmtojpeg**): To dump (interpret) an EXIF header: Exifdump ((http://topo.math.u-psud.fr/˜ bousch/exifdump.py) or Jhead (http://www.sentex.net/˜ mwandel/jhead).

A Python EXIF library and dumper: http://pyexif.sourceforge.net.

Latex2html converts Latex document source to HTML document source. Part of that involves graphics, and Latex2html uses Netpbm tools for some of that. But Latex2html through its history has had some rather esoteric codependencies with Netpbm. Older Latex2html doesn't work with current Netpbm. Latex2html-99.2beta8 works, though.

Other Graphics Formats

People never seem to tire of inventing new graphics formats, often completely redundant with preexisting ones. Netpbm cannot keep up with them. Here is a list of a few that we know Netpbm does *not* handle (yet).

CAL (originated by US Department Of Defense, favored by architects). http://www.landfield.com/faqs/graphics/fileformats-faq/part3/section-24.html

array formats dx, general, netcdf, CDF, hdf, cm

CGM+

Windows Meta File (.WMF). Libwmf converts from WMF to things like Latex, PDF, PNG. Some of these can be input to Netpbm.

Microsoft Word, RTF. Microsoft keeps a proprietary hold on these formats. Any software you see that can handle them is likely to cost money.

DXF (AutoCAD)

HISTORY

Netpbm has a long history, starting with Jef Poskanzer's **Pbmplus** package in 1988. The file *HISTORY* in the Netpbm source code contains a historical overview as well as a detailed history release by release.

AUTHOR

Netpbm is based on the **Pbmplus** package by Jef Poskanzer, first distributed in 1988 and maintained by him until 1991. But the package contains work by countless other authors, added since Jef's original work. In fact, the name is derived from the fact that the work was contributed by people all over the world via the Internet, when such collaboration was still novel enough to merit naming the package after it.

Bryan Henderson has been maintaining **Netpbm** since 1999. In addition to packaging work by others, Bryan has also written a significant amount of new material for the package.

NAME

palmtopnm - convert a Palm pixmap into a portable anymap

SYNOPSIS

palmtopnm [**-verbose**] [**-rendition** *N*] [**-showhist**]
[**-forceplain**] [*pnmfile*]
palmtopnm -transparent [**-verbose**] [*pnmfile*]

DESCRIPTION

Reads a Palm pixmap as input, from stdin or *pnmfile*. Produces either a portable pixmap as output, or writes the value of the transparent color in the Palm pixmap to stdout.

OPTIONS

-verbose
Display various interesting information about the input file and process.

-transparent
If the Palm pixmap has a transparent color set, the RGB value for that color will be written to stdout as in the form `#RRGGBB`, where `RR`, `GG`, and `BB` are two-digit hexadecimal numbers indicating a value between 0 and 255. If no transparent color is set in the bitmap, nothing will be output. No additional output will be generated; no anymap will be output.

-rendition N
Palm pixmaps may contain several different renditions of the same pixmap, with different depths. By default, **palmtopnm** operates on the first rendition (rendition number 1) in the pixmap. This switch allows you to operate on a different rendition. The value must be between 1 and the number of renditions in the pixmap, inclusive.

-showhist
Writes a histogram of colors in the input file to stderr.

-forceplain
Force the output anymap to be in ASCII 'plain' netpbm format.

SEE ALSO

pnmtopalm(1), **pnm(5)**

BUGS

An additional compression format, "packbits," has been added with PalmOS 4.0. This package should be updated to handle it.

You currently cannot generate an alpha mask if the Palm pixmap has a transparent color. However, you can still do this with **ppmcolormask** with a Netpbm pipe similar to:

palmtopnm pixmap.palm | ppmcolormask ‘palmtopnm -transparent pixmap.palm‘

AUTHORS

This program was originally written as Tbmptopnm.c, by Ian Goldberg. It was heavily modified by Bill Janssen to add color, compression, and transparency function.
Copyright 1995-2001 by Ian Goldberg and Bill Janssen.

NAME

pamchannel - extract channels from a PAM image

SYNOPSIS

pamchannel [**-infile** *infile*] [*channum*] ...

All options may be abbreviated to the shortest unique prefix.

DESCRIPTION

Reads a PAM or PNM image as input. Produces a PAM image as output, consisting of the indicated channels of the input.

The output is the same dimensions as the input, except that the depth is the number of *channum* arguments you supply. The tuple type is a null string.

OPTIONS

-infile *infile*

This specifies the input file, which defaults to Standard Input. You may specify **-** to select Standard Input explicitly.

This is a little unconventional for Netpbm programs, which usually have the input file specification as an argument. For **pamchannel**, the arguments are channel numbers.

SEE ALSO

pnm(5)

NAME

pamcut - cut a rectangle out of a PAM, PBM, PGM, or PPM image

SYNOPSIS

pamcut [**-left** *leftcol*] [**-right** *rightcol*] [**-top** *toprow*] [**-bottom** *bottomrow*] [**-width** *width*] [**-height** *height*] [**-pad**] [**-verbose**] [*left right width height*] [*pnmfi le*]

All options may be abbreviated to the shortest unique prefi x.

DESCRIPTION

Reads a PAM, PBM, PGM, or PPM image as input. Extracts the specifi ed rectangle, and produces the same kind of image as output.

There are two ways to specify the rectangle to cut: arguments and options. Options are easier to remember and read, more expressive, and allow you to use defaults. Arguments were the only way available before July 2000.

If you use both options and arguments, the two specifi cations get mixed in an unspecifi ed way.

To use options, just code any mixture of the **-left**, **-right**, **-top**, **-bottom**, **-width**, and **-height** options. What you don't specify defaults. It is an error to overspecify, i.e. to specify all three of **-left**, **-right**, and **-width** or **-top**, **-bottom**, and **-height**.

To use arguments, specify all four of the *left*, *right*, *width*, and *height* arguments. *left* and *top* have the same effect as specifying them as the argument of a **-left** or **-top** option, respectively. *width* and *height* have the same effect as specifying them as the argument of a **-width** or **-height** option, respectively, where they are positive. Where they are not positive, they have the same effect as specifying one less than the value as the argument to a **-right** or **-bottom** option, respectively. (E.g. *width* = 0 makes the cut go all the way to the right edge). Before July 2000, negative numbers were not allowed for *width* and *height*.

Input is from Standard Input if you don't specify the input fi le *pnmfi le*.

Output is to Standard Output.

If you are splitting a single image into multiple same-size images, **pamdice** is faster than running **pamcut** multiple times.

OPTIONS

-left The column number of the leftmost column to be in the output. If a nonnegative number, it refers to columns numbered from 0 at the left, increasing to the right. If negative, it refers to columns numbered -1 at the right, decreasing to the left.

-right The column number of the rightmost column to be in the output, numbered the same as for **-left.**

-top The row number of the topmost row to be in the output. If a nonnegative number it refers to rows numbered from 0 at the top, increasing downward. If negative, it refers to columns numbered -1 at the bottom, decreasing upward.

-bottom
The row number of the bottom-most row to be in the output, numbered the same as for **-top**.

-width The number of columns to be in the output. Must be positive.

-height The number of rows to be in the output. Must be positive.

-pad If the rectangle you specify is not entirely within the input image, **pamcut** fails unless you also specify **-pad**. In that case, it pads the output with black up to the edges you specify. You can use this option if you need to have an image of certain dimensions and have an image of arbitrary dimensions.

pnmpad also adds borders to an image, but you specify their width directly.

-verbose
Print information about the processing to Standard Error.

SEE ALSO

pnmcrop(1), **pnmpad**(1), **pnmcat**(1), **pgmslice**(1), **pnm**(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pamdeinterlace − remove ever other row from a PAM/PNM image

SYNOPSIS

pamdeinterlace [**-takeodd**] [**-takeeven**] *N* [*infi le*]

You can use the minimum unique abbreviation of the options. You can use two hyphens instead of one. You can separate an option name from its value with white space instead of an equals sign.

DESCRIPTION

pamdeinterlace Removes all the even-numbered or odd-numbered rows from the input PNM or PAM image. Specify which with the **-takeeven** and **-takeodd** options.

This can be useful if the image is a video capture from an interlaced video source. In that case, each row shows the subject 1/60 second before or after the two rows that surround it. If the subject is moving, this can detract from the quality of the image.

Because the resulting image is half the height of the input image, you will then want to use **pamstretch** or **pnmscale** to restore it to its normal height:

```
pamdeinterlace myimage.ppm | pamstretch -yscale=2 >newimage.ppm
```

OPTIONS

-takeodd

Take the odd-numbered rows from the input and put them in the output. The rows are numbered starting at zero, so the fi rst row in the output is the second row from the input. You cannot specify both **-takeeven** and **-takeodd**.

-takeeven

Take the even-numbered rows from the input and put them in the output. The rows are numbered starting at zero, so the fi rst row in the output is the fi rst row from the input. This is the default. You cannot specify both **-takeeven** and **-takeodd**.

SEE ALSO

pamstretch(1), **pnmscale**(1)

NAME

pamdice - slice a Netpbm image into many horizontally and/or vertically

SYNOPSIS

pamslice -outstem= *fi lenamestem* [**-width=***width*] [**-height=***height*] [**-verbose**] [*fi lename*]

You can use the minimum unique abbreviation of the options. You can use two hyphens instead of one. You can separate an option name from its value with white space instead of an equals sign.

DESCRIPTION

Reads a PAM, PBM, PGM, or PPM image as input. Splits it horizontally and/or vertically into equal size pieces and writes them into separate fi les as the same kind of image.

See the **-outstem** option for information on naming of the output fi les.

The **-width** and **-height** options determine the size of the output pieces.

pnmcat can rejoin the images.

OPTIONS

-outstem=fi lenamestem

This option determines the names of the output fi les. Each output fi le is named *fi le-namestem_y_x***.***type* where *fi lenamestem* is the value of the **-outstem** option, *x* and y are the horizontal and vertical locations, respectively, in the input image of the output image, zero being the leftmost and top, and *type* is **.pbm**, **.pgm**, **.ppm**, or **.pam**, depending on the type of image.

-width=*width*

gives the width in pixels of the output images. The rightmost pieces are smaller than this if the input image is not a multiple of *width* pixels wide.

-height=*height*

gives the height in pixels of the output images. The bottom pieces are smaller than this if the input image is not a multiple of *height* pixels high.

-verbose

Print information about the processing to Standard Error.

SEE ALSO

pamcut(1), **pnmcat**(1), **pgmslice**(1), **pnm**(5)

NAME

pamfile - describe a Netpbm (PAM or PNM) file

SYNOPSIS

pamfile [**-allimages**] [*file* ...]

DESCRIPTION

Reads one or more Netpbm files as input. Writes out short descriptions of the image type, size, etc. This is mostly for use in shell scripts, so the format is not particularly pretty.

OPTIONS

-allimages

Describe every image in each input file. Without this option, **pamfile** describes only the first image in each input file. Note that before July 2000, a file could not contain more than one image and many programs ignore all but the first.

SEE ALSO

pam(5), **file**(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pamoil - turn a PAM image into an oil painting

SYNOPSIS

pamoil [**-n** *N*] [*pamfi le*]

DESCRIPTION

Reads a Netpbm image as input. Does an "oil transfer", and writes the same type of Netpbm image as output.

The oil transfer is described in "Beyond Photography" by Holzmann, chapter 4, photo 7. It's a sort of localized smearing.

The smearing works like this: First, assume a grayscale image. For each pixel in the image, **pamoil** looks at a square neighborhood around it. **pamoil** determines what is the most common pixel intensity in the neighborhood, and puts a pixel of that intensity into the output in the same position as the input pixel.

For color images, or any arbitrary multi-channel image, **pamoil** computes each channel (e.g. red, green, and blue) separately the same way as the grayscale case above.

At the edges of the image, where the regular neighborhood would run off the edge of the image, **pamoil** uses a clipped neighborhood.

OPTIONS

-n *size* This is the size of the neighborhood used in the smearing. The neighborhood is this many pixels in all four directions.

The default is 3.

SEE ALSO

pgmbentley(1), **ppmrelief**(1), **ppm**(5)

AUTHOR

Based on pgmoil Copyright (C) 1990 by Wilson Bent (whb@hoh-2.att.com)

Modifi ed to ppm by Chris Sheppard, June 25, 2001

Modifi ed to pnm, using pam functions, by Bryan Henderson June 28, 2001.

NAME

pamstretch-gen − use pamstretch and pnmscale to scale by non-integer values

SYNOPSIS

pamstretch-gen *N* [*pnmfi le*]

DESCRIPTION

pamstretch-gen is a program which uses **pamstretch**(1), **pnmfi le**(1), and **pnmscale**(1) to smoothly scale up a PNM fi le by any ratio; it's like a more general version of pamstretch (hence the name). But other than the 'any ratio' bit, it's much the same as pamstretch. :-)

BUGS

Uses awk just to make some simple fboating-point calculations, which is probably overkill. But using dc makes my head hurt.

SEE ALSO

pamstretch(1), **pnmscale**(1)

AUTHOR

Russell Marks (russell.marks@ntlworld.com).

NAME

pamstretch − scale up a PNM or PAM image by interpolating between pixels.

SYNOPSIS

pamstretch [**-xscale=***X*] [**-yscale=***Y*]
[**-blackedge**] [**-dropedge**] *N* [*infi le*]

You can use the minimum unique abbreviation of the options. You can use two hyphens instead of one. You can separate an option name from its value with white space instead of an equals sign.

DESCRIPTION

pamstretch scales up pictures by integer values, either vertically, horizontally, or both. **pamstretch** differs from **pnmscale** and **pnmenlarge** in that when it inserts the additional rows and columns, instead of making the new row or column a copy of its neighbor, **pamstretch** makes the new row or column an interpolation between its neighbors. In some images, this produces better looking output.

To scale up to non-integer pixel sizes, e.g. 2.5, try **pamstretch-gen**(1) instead.

Options let you select alternative methods of dealing with the right/bottom edges of the picture. Since the interpolation is done between the top-left corners of the scaled-up pixels, it's not obvious what to do with the right/bottom edges. The default behaviour is to scale those up without interpolation (more precisely, the right edge is only interpolated vertically, and the bottom edge is only interpolated horizontally), but there are two other possibilities, selected by the **blackedge** and **dropedge** options.

PARAMETERS

The *N* parameter is the scale factor. It is valid only if you *don't* specify **-xscale** or **-yscale**. In that case, **pamstretch** scales in both dimensions and by the scale factor *N*.

OPTIONS

-xscale=*X*
: This is the horizontal scale factor. If you don't specify this, but do specify a vertical scale factor, the horizontal scale factor is 1.

-yscale=*Y*
: This is the vertical scale factor. If you don't specify this, but do specify a horizontal scale factor, the vertical scale factor is 1.

-blackedge
: interpolate to black at right/bottom edges.

-dropedge
: drop one (source) pixel at right/bottom edges. This is arguably more logical than the default behaviour, but it means producing output which is a slightly odd size.

BUGS

Usually produces fairly ugly output for PBMs. For most PBM input you'll probably want to reduce the 'noise' fi rst using something like **pnmnlfi lt**(1).

SEE ALSO

pamstretch-gen(1), **pnmenlarge**(1), **pnmscale**(1), **pnmnlfi lt**(1)

AUTHOR

Russell Marks (russell.marks@ntlworld.com).

NAME

pamtopnm - convert PAM image to PBM, PGM, or PPM

SYNOPSIS

pamtopnm [**-assume**] [*pnmfile*]

All options may be abbreviated to the shortest unique prefix.

DESCRIPTION

Reads a PAM image as input. Produces an equivalent PBM, PGM, or PPM (i.e. PNM) image, whichever is most appropriate, as output.

pamtopnm assumes the PAM image represents the information required for a PBM, PGM, or PPM image if its tuple type is "BLACKANDWHITE", "GRAYSCALE", or "RGB" and its depth and maxval are appropriate. If this is not the case, **pamtopnm** fails.

However, you can override the tuple type requirement with the **-assume** option.

As with any Netpbm program that reads PAM images, **pamtopnm** also reads PNM images as if they were PAM. In that case, **pamtopnm**'s functions reduces to simply copying the input to the output. But this can be useful in a program that doesn't know whether its input is PAM or PNM but needs to feed it to a program that only recognizes PNM.

OPTIONS

-assume

When you specify **-assume**, you tell **pamtopnm** that you personally vouch for the fact that the tuples contain the same data as belongs in the channels of a PBM, PGM, or PPM file. The depth must still conform, though, so to truly force a conversion, you may have to run the input through **pamchannel** first. But be careful with **-assume**. When you -assume, you make an -ass of u and me.

SEE ALSO

pbmtopgm(1), **pgmtopbm**(1), **pgmtoppm**(1), **ppmtopgm**(1), **pam**(5), **pnm**(5), **pbm**(5), **pgm**(5), **ppm**(5)

NAME

pbmclean - flip isolated pixels in portable bitmap

SYNOPSIS

pbmclean [**-minneighbors=***N*] [**-black**|**-white**] [*pbmfile*]

You can use the minimum unique abbreviation of the options. You can use two hyphens instead of one. You can separate an option name from its value with white space instead of an equals sign.

Before December 2001, **pbmclean** accepted **-***N* instead of **-minneighbors**.

DESCRIPTION

pbmclean cleans up a PBM image of random specs. It reads a PBM image as input and outputs a PBM that is the same as the input except with every pixel which has less than *N* identical neighbours inverted.

The default for *N* is 1 - only completely isolated pixels are flipped.

(A value of *N* greater than 8 generates a completely inverted image (but use **pnminvert** to do that) -- or a completely white or completely black image with the **-black** or **-white** option).

pbmclean considers the area beyond the edges of the image to be white. (This matters when you consider pixels right on the edge of the image).

You can use **pbmclean** to clean up "snow" on bitmap images.

OPTIONS

-black

-white Flip pixels of the specified color. By default, if you specify neither **-black** nor **-white**, **pbmclean** flips both black and white pixels which do not have sufficient identical neighbors. If you specify **-black**, **pbmclean** leaves the white pixels alone and just erases isolated black pixels. Vice versa for **-white**. You may specify both **-black** and **-white** to get the same as the default behavior.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1990 by Angus Duggan Copyright (C) 1989 by Jef Poskanzer. Copyright (C) 2001 by Michael Sternberg.

NAME

pbmlife - apply Conway's rules of Life to a portable bitmap

SYNOPSIS

pbmlife [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Applies the rules of Life to it for one generation, and produces a portable bitmap as output.

A white pixel in the image is interpreted as a live beastie, and a black pixel as an empty space.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988, 1991 by Jef Poskanzer.

NAME

pbmmake - create a blank bitmap of a specified size

SYNOPSIS

pbmmake [**-white**|**-black**|**-gray**] *width height*

DESCRIPTION

Produces a portable bitmap of the specified width and height. The color defaults to white.

OPTIONS

In addition to the usual **-white** and **-black**, this program implements **-gray**. This gives a simple 50% gray pattern with 1's and 0's alternating.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pbm(5), ppmmake(1)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pbmmask - create a mask bitmap from a regular bitmap

SYNOPSIS

pbmmask [**-expand**] [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Creates a corresponding mask bitmap and writes it out.

The color to be interpreted as "background" is determined automatically. Regardless of which color is background, the mask will be white where the background is and black where the figure is.

This lets you do a masked paste like this, for objects with a black background:

```
  pbmmask obj > objmask
  pnmpaste < dest -and objmask <x> <y> | pnmpaste -or obj <x> <y>
```

For objects with a white background, you can either invert them or add a step:

```
  pbmmask obj > objmask
  pnminvert objmask | pnmpaste -and obj 0 0 > blackback
  pnmpaste < dest -and objmask <x> <y> | pnmpaste -or blackback <x> <y>
```

Note that this three-step version works for objects with black backgrounds too, if you don't care about the wasted time.

You can also use masks with graymaps and pixmaps, using the *pnmarith* tool. For instance:

```
  ppmtopgm obj.ppm | pgmtopbm -threshold | pbmmask > objmask.pbm
  pnmarith -multiply dest.ppm objmask.pbm > t1.ppm
  pnminvert objmask.pbm | pnmarith -multiply obj.ppm - > t2.ppm
  pnmarith -add t1.ppm t2.ppm
```

An interesting variation on this is to pipe the mask through the *pnmsmooth* script before using it. This makes the boundary between the two images less sharp.

OPTIONS

-expand

Expands the mask by one pixel out from the image. This is useful if you want a little white border around your image. (A better solution might be to turn the *pbmlife* tool into a general cellular automaton tool...)

SEE ALSO

ppmcolormask(1), **pnmpaste**(1), **pnminvert**(1), **pbm**(5), **pnmarith**(1), **pnmsmooth**(1)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pbmpage - create a one page test pattern for printing

SYNOPSIS

pbmpage [**-a4**] *test_pattern*

DESCRIPTION

pbmpage generates a one page test pattern to print on a sheet of paper, for use in calibrating a printer. The test pattern in is PBM format.

pbmpage produces an image intended for 600 dots per inch printer resolution.

If you are printing on an HP PPA printer, you can convert the output of this program to a stream that you can feed to the printer with **pbmtoppa**.

Bear in mind that when you print the test pattern, you are testing not only the printer, but any converter or driver software along the printing path. Any one of these components may adjust margins, crop the image, erase edges, and such.

If, due to addition of margins, the printer refuses to print the image because it is too big, use **pnmcut** to cut the right and bottom edges off the test pattern until it is small enough to print.

test_pattern is the number of the test pattern to generate, as follows. The default is **1**.

1 A grid ruled in numbers of pixels. The outermost rules are exactly at the edges of the paper.

2 A vertical line segment, one pixel wide, extending 1/2" up from the exact center of the page.

3 Two diagonal line segments, one starting at the upper left corner of the page, the other starting from the lower left corner of the page. Both extend 1/2" toward the center of the page at 45 degrees.

OPTIONS

-a4 Generate an image for A4 (European) paper. Without this option, **pbmpage** generates an image for US standard paper (8 1/2" wide x 11" high).

SEE ALSO

pbmtoppa(1), **pnmcut**(1), **pbm**(5)

AUTHOR

Tim Norman. Copyright (C) 1998. Licensed under GNU Public License

Manual page by Bryan Henderson, May 2000.

NAME

pbmpscale - enlarge a portable bitmap with edge smoothing

SYNOPSIS

pbmpscale N [pbmfile]

DESCRIPTION

Reads a portable bitmap as input, and outputs a portable bitmap enlarged N times. Enlargement is done by pixel replication, with some additional smoothing of corners and edges.

SEE ALSO

pnmenlarge(1), ppmscale(1), pbm(5)

AUTHOR

Copyright (C) 1990 by Angus Duggan Copyright (C) 1989 by Jef Poskanzer.

NOTES

pbmpscale works best for enlargements of 2. Enlargements greater than 2 should be done by as many enlargements of 2 as possible, followed by an enlargement by the remaining factor.

NAME

pbmreduce - read a portable bitmap and reduce it N times

SYNOPSIS

pbmreduce [**-fbyd**|**-fs**|**-threshold**] [**-value** *val*] *N* [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Reduces it by a factor of *N*, and produces a portable bitmap as output.

pbmreduce duplicates a lot of the functionality of *pgmtopbm;* you could do something like **pnmscale | pgmtopbm,** but *pbmreduce* is a lot faster.

pbmreduce can be used to "re-halftone" an image. Let's say you have a scanner that only produces black&white, not grayscale, and it does a terrible job of halftoning (most b&w scanners fit this description). One way to fix the halftoning is to scan at the highest possible resolution, say 300 dpi, and then reduce by a factor of three or so using *pbmreduce*. You can even correct the brightness of an image, by using the **-value** flag.

OPTIONS

By default, the halftoning after the reduction is done via boustrophedonic Floyd-Steinberg error diffusion; however, the **-threshold** flag can be used to specify simple thresholding. This gives better results when reducing line drawings.

The **-value** flag alters the thresholding value for all quantizations. It should be a real number between 0 and 1. Above 0.5 means darker images; below 0.5 means lighter.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmenlarge(1), pnmscale(1), pgmtopbm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pbmtext - render text into a bitmap

SYNOPSIS

pbmtext [**−font** *fontfi le*] [**−builtin** *fontname*] [**−space** *pixels*] [*text*]

DESCRIPTION

Takes the specifi ed text, either a single line from the command line or multiple lines from standard input, and renders it into a bitmap.

In the bitmap, each line of input is a line of output. Formatting characters such as newline have no effect on the formatting; like any unprintable character, they turn into spaces.

The bitmap is just wide enough for the longest line of text, plus margins, and just high enough to contain the lines of text, plus margins. The left and right margins are twice the width of the widest character in the font; the top and bottom margins are the height of the tallest character in the font. But if the text is only one line, all the margins are half of this.

OPTIONS

-font,**-builtin**

By default, pbmtext uses a built-in font called bdf (about a 10 point Times-Roman font). You can use a fi xed width font by specifying **-builtin fi xed**.

You can also specify your own font with the **-font** flag. The *fontfi le* is either a BDF fi le from the X window system or a PBM fi le.

If the *fontfi le* is a PBM fi le, it is created in a very specifi c way. In your window system of choice, display the following text in the desired (fi xed-width) font:

```
M ",/^_[‘jpqy| M

/  !"#$%&’()*+ /
< ,-./01234567 <
> 89:;<=>?@ABC >
@ DEFGHIJKLMNO @
_ PQRSTUVWXYZ[ _
{ \]^_‘abcdefg {
} hijklmnopqrs }
~  tuvwxyz{|}~   ~

M ",/^_[‘jpqy| M
```

Do a screen grab or window dump of that text, using for instance **xwd**, **xgrabsc**, or **screendump**. Convert the result into a pbm fi le. If necessary, use **pnmcut** to remove everything except the text. Finally, run it through **pnmcrop** to make sure the edges are right up against the text. **pbmtext** can fi gure out the sizes and spacings from that.

-space *pixels*

Add *pixels* pixels of space between characters. This is in addition to whatever space surrounding characters is built into the font, which is usually enough to produce a reasonable string of text.

pixels may be negative to crowd text together, but the author has not put much thought or testing into how this works in every possible case, so it might cause disastrous results.

USAGE

Often, you want to place text over another image. One way to do this is with **ppmlabel**. **ppmlabel** does not give you the font options that **pbmtext** does, though.

Another way is to use **pbmtext** to create an image containing the text, then use **pnmcomp** to overlay the text image onto your base image. To make only the text (and not the entire rectangle containing it) cover the base image, you will need to give **pnmcomp** a mask, via its **-alpha** option. You can just use the text image itself as the mask, as long as you also specify the **-invert** option to **pnmcomp**.

If you want to overlay colored text instead of black, just use **ppmchange** to change all black pixels to the color of your choice before overlaying the text image. But still use the original black and white image for the alpha mask.

If you want the text at an angle, use **pnmrotate** on the text image (and alpha mask) before overlaying.

SEE ALSO

pnmcut(1), **pnmcrop**(1), **pnmcomp**(1), **ppmchange**(1), **pnmrotate**(1), **ppmlabel**(1), **pbm**(5)

AUTHOR

Copyright (C) 1993 by Jef Poskanzer and George Phillips

NAME

pbmto10x - convert a portable bitmap into Gemini 10X printer graphics

SYNOPSIS

pbmto10x [**-h**] [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a file of Gemini 10X printer graphics as output. The 10x's printer codes are alleged to be similar to the Epson codes.

Note that there is no 10xtopbm tool - this transformation is one way.

OPTIONS

The resolution is normally 60H by 72V. If the **-h** flag is specified, resolution is 120H by 144V. You may find it useful to rotate landscape images before printing.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1990 by Ken Yap

NAME

pbmto4425 − Display PBM images on an AT&T 4425 terminal

SYNOPSIS

pbmto4425 [*pbmfile*]

DESCRIPTION

Pbmto4425 displays PBM format images on an AT&T 4425 ASCII terminal using that terminal's mosaic graphics character set. The program should also work with other VT100-like terminals with mosaic graphics character sets such as the C. Itoh CIT-101, but it has not yet been tested on terminals other than the 4425.

Pbmto4425 puts the terminal into 132 column mode to achieve the maximum resolution of the terminal. In this mode the terminal has a resolution of 264 columns by 69 rows. The pixels have an aspect ratio of 1:2.6, therefore an image should be processed before being displayed in a manner such as this:

```
% pnmscale −xscale 2.6 pnmfile \
        | pnmscale −xysize 264 69 \
        | ppmtopgm \
        | pgmtopbm \
        | pbmto4425
```

AUTHOR

Copyright (C) 1993 by Robert Perlberg

NAME

pbmtoascii - convert a portable bitmap into ASCII graphics

SYNOPSIS

pbmtoascii [**-1x2**|**-2x4**] [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a somewhat crude ASCII graphic as output.

Note that there is no asciitopbm tool - this transformation is one-way.

OPTIONS

The **-1x2** and **-2x4** flags give you two alternate ways for the bits to get mapped to characters. With **1x2**, the default, each character represents a group of 1 bit across by 2 bits down. With **-2x4**, each character represents 2 bits across by 4 bits down. With the 1x2 mode you can see the individual bits, so it's useful for previewing small bitmaps on a non-graphics terminal. The 2x4 mode lets you display larger bitmaps on a standard 80-column display, but it obscures bit-level details. 2x4 mode is also good for displaying graymaps - "pnmscale -width 158 | pgmnorm | pgmtopbm -thresh" should give good results.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988, 1992 by Jef Poskanzer.

NAME

pbmtoatk - convert portable bitmap to Andrew Toolkit raster object

SYNOPSIS

pbmtoatk [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a Andrew Toolkit raster object as output.

SEE ALSO

atktopbm(1), pbm(5)

AUTHOR

Copyright (C) 1991 by Bill Janssen.

NAME

pbmtobg - convert a portable bitmap into BitGraph graphics

SYNOPSIS

pbmtobg [*rasterop*] [*x y*] < *pbmfile*

DESCRIPTION

Reads a portable bitmap as input. Produces BBN BitGraph terminal Display Pixel Data (DPD) sequence as output.

The rasterop can be specified on the command line. If this is omitted, 3 (replace) will be used. A position in (x,y) coordinates can also be specified. If both are given, the rasterop comes first. The portable bitmap is always taken from the standard input.

Note that there is no bgtopbm tool.

SEE ALSO

pbm(5)

AUTHOR

Copyright 1989 by Mike Parker.

NAME

pbmtocmuwm - convert a portable bitmap into a CMU window manager bitmap

SYNOPSIS

pbmtocmuwm [*pbmfi le*]

DESCRIPTION

Reads a portable bitmap as input. Produces a CMU window manager bitmap as output.

SEE ALSO

cmuwmtopbm(1), pbm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pbmtoepsi - convert a portable bitmap into an encapsulated PostScript style preview bitmap

SYNOPSIS

pbmtoepsi [**-bbonly**] [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produce an encapsulated Postscript style bitmap as output. The output is not a stand alone postscript file, it is only a preview bitmap, which can be included in an encapsulated PostScript file. Note that there is no epsitopbm tool - this transformation is one way.

This utility is a part of the pstoepsi tool by Doug Crabill (dgc@cs.purdue.edu).

OPTIONS

-bbonly
Only create a boundary box, don't fill it with the image.

SEE ALSO

pbm(5), **pnmtops**(1), **pstopnm**(1), **psidtopgm**(1), **pbmtolps**(1), **ps**(1)

AUTHOR

Copyright (C) 1988 Jef Poskanzer, modified by Doug Crabill 1992

NAME

pbmtoepson - convert a portable bitmap into Epson printer graphics

SYNOPSIS

pbmtoepson [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a file of Epson printer graphics as output.

Note that there is no epsontopbm tool - this transformation is one way.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1991 by John Tiller (tiller@galois.msfc.nasa.gov) and Jef Poskanzer.

NAME
pbmtog3 - convert a portable bitmap into a Group 3 fax fi le

SYNOPSIS
pbmtog3 [*pbmfi le*]

DESCRIPTION
Reads a portable bitmap as output. Produces a Group 3 fax fi le as input.

REFERENCES
The standard for Group 3 fax is defi ned in CCITT Recommendation T.4.

BUGS
Probably.

SEE ALSO
g3topbm(1), pbm(5)

AUTHOR
Copyright (C) 1989 by Paul Haeberli <paul@manray.sgi.com>.

NAME

pbmtogem - convert a portable bitmap into a GEM .img file

SYNOPSIS

pbmtogem [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a compressed GEM .img file as output.

BUGS

pbmtogem does not support compression of repeated lines

SEE ALSO

gemtopbm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by David Beckemeyer (bdt!david) and Jef Poskanzer.

NAME
pbmtogo - convert a portable bitmap into compressed GraphOn graphics

SYNOPSIS
pbmtogo [*pbmfi le*]

DESCRIPTION
Reads a portable bitmap as input. Produces 2D compressed GraphOn graphics as output. Be sure to set up your GraphOn with the following modes: 8 bits / no parity; obeys no XON/XOFF; NULs are accepted. These are all on the Comm menu. Also, remember to turn off tty post processing. Note that there is no gotopbm tool.

SEE ALSO
pbm(5)

AUTHOR
Copyright (C) 1988, 1989 by Jef Poskanzer, Michael Haberler, and Bo Thide'.

NAME

pbmtoicon - convert a portable bitmap into a Sun icon

SYNOPSIS

pbmtoicon [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a Sun icon as output.

SEE ALSO

icontopbm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pbmtolj - convert a PBM image to HP LaserJet format

SYNOPSIS

pbmtolj [**-resolution** *N*] [**-fbat**] [**-noreset**] [**-packbits**] [**-delta**] [**-compress**] [*pbmfi le*] [**-copies** *N*]

DESCRIPTION

Reads a PBM image as input. Produces HP LaserJet data as output.

Note that there is no ljtopbm tool.

OPTIONS

-resolution
Specifi es the resolution of the output device, in dpi. Typical values are 75, 100, 150, 300, and 600. The default is 75.

-fbat Suppresses positioning information. The default is to write the sequence *ESC & l 0 E* to the output fi le.

-noreset
Prevents pbmtolj from writing the reset sequences to the beginning and end of the output fi le.

-packbits
Enables use of TIFF packbits compression.

-delta Enables use of delta-between-rows compression.

-compress
Enables use of both TIFF packbits, and delta-between-rows compression.

-copies Specifi es the the number of copies. The default is 1. This option controls the "number of copies" printer control; **pbmtolj** generates only one copy of the image.

All fbgs can be abbreviated to their shortest unique prefi x.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer and Michael Haberler. **-fbat** and **-noreset** options added by Wim Lewis. **-delta, -packbits,** and **-compress** options added by Dave Platt.

NAME
pbmtoln03 - convert protable bitmap to DEC LN03+ Sixel output

SYNOPSIS
pbmtoln03 [**-rltbf**] *pbmfile*

DESCRIPTION
Reads a portable bitmap as input. Produces a DEC LN03+ Sixel output file.

OPTIONS

-l nn	Use "nn" as value for left margin (default 0).
-r nn	Use "nn" as value for right margin (default 2400).
-t nn	Use "nn" as value for top margin (default 0).
-b nn	Use "nn" as value for bottom margin (default 3400).
-f nn	Use "nn" as value for form length (default 3400).

SEE ALSO
pbm(5)

AUTHOR
Tim Cook, 26 Feb 1992

NAME

pbmtolps - convert portable bitmap to PostScript

SYNOPSIS

pbmtolps [-dpi n] [pbmfile]

DESCRIPTION

Reads a portable bitmap as input, and outputs PostScript. The output Postscript uses lines instead of the image operator to generate a (device dependent) picture which will be imaged much faster.

The Postscript path length is constrained to be less that 1000 points so that no limits are overrun on the Apple Laserwriter and (presumably) no other printers.

SEE ALSO

pbm(5), **pnmtops**(1), **pstopnm**(1), **pbmtoepsi**(1), **psidtopgm**(1), **gs**(1)

AUTHOR

George Phillips <phillips@cs.ubc.ca>

NAME

pbmtomacp - convert a portable bitmap into a MacPaint file

SYNOPSIS

pbmtomacp [**-l** *left*] [**-r** *right*] [**-b** *bottom*] [**-t** *top*] [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. If no input-file is given, standard input is assumed. Produces a MacPaint file as output.

The generated file is only the data fork of a picture. You will need a program such as *mcvert* to generate a Macbinary or a BinHex file that contains the necessary information to identify the file as a PNTG file to MacOS.

OPTIONS

Left, right, bottom & top let you define a square into the pbm file, that must be converted. Default is the whole file. If the file is too large for a MacPaint-file, the bitmap is cut to fit from (left, top).

BUGS

The source code contains comments in a language other than English.

SEE ALSO

ppmtopict(1), macptopbm(1), pbm(5), mcvert(1)

AUTHOR

Copyright (C) 1988 by Douwe van der Schaaf (...!mcvax!uvapsy!vdschaaf).

NAME

pbmtomda - convert a portable bitmap to a Microdesign .mda

SYNOPSIS

pbmtomda [**-d**][**-i**][**--**] *[pbmfile]*

DESCRIPTION

Reads a portable bitmap file as input. Reads from stdin if input file is omitted. Produces a MicroDesign 2 area file (.MDA) as output.

OPTIONS

-d Halve the height of the output file, to compensate for the aspect ratio used in MicroDesign files.

-i Invert the colours used.

-- End of options (use this if the filename starts with "-")

BUGS

There's no way to produce files in MicroDesign 3 format. MD3 itself and mdatopbm(1) can read files in either format.

SEE ALSO

mdatopbm(1), pbm(5)

AUTHOR

Copyright (C) 1999 John Elliott <jce@seasip.demon.co.uk>.

NAME
pbmtomgr - convert a portable bitmap into a MGR bitmap

SYNOPSIS
pbmtomgr [*pbmfile*]

DESCRIPTION
Reads a portable bitmap as input. Produces a MGR bitmap as output.

SEE ALSO
mgrtopbm(1), pbm(5)

AUTHOR
Copyright (C) 1989 by Jef Poskanzer.

NAME

pbmtonokia - convert a portable bitmap to Nokia Smart Messaging Formats

SYNOPSIS

pbmtonokia [options] [*pbmfi le*]

DESCRIPTION

Reads a portable bitmap as input. Produces a Nokia Smart Messaging (hexcode, .nok, .ngg) fi le as output.

OPTIONS

-fmt Specifi es the output format (default is HEX_NOL).

HEX_NOL
Nokia Operator Logo as (uploadable) hexcode. Use option -net to specify network code.

HEX_NGG
Nokia Group Graphic as (uploadable) hexcode.

HEX_NMP
Nokia Picture Message as (uploadable) hexcode. Use option -txt to specify optional text message.

NOL Nokia Operator Logo as .nol format. This is editable by the Group-Graphic Editor from Kessler Wireless Design (www.kessler-design.com)

NGG Nokia Group Graphic as .ngg format. This is editable by the Group-Graphic Editor from Kessler Wireless Design (www.kessler-design.com)

-net Specifi es the 6 hex-digit operator network code for Operator Logos (Default is 62F210 = D1,Germany).

-txt Specifi es the text message for Picture Messages. Default is no text message.

LIMITATIONS

Currently limited to rows<=255 and columns<=255. Supports only b/w graphics, not animated.

SEE ALSO

pbm(5), **Nokia Smart Messaging Specifi cation (http://forum.nokia.com)**

AUTHOR

Copyright (C) 2001 Tim Ruehsen <tim.ruehsen@openmediasystem.de>.

NAME

pbmtopgm - convert PBM image to PGM by averaging areas

SYNOPSIS

pbmtopgm *width height* [*pbmfi le*]

DESCRIPTION

pbmtopgm reads a portable bitmap as input. It outputs a portable graymap in which each pixel's gray level is the average the surrounding black and white input pixels. The surrounding area is a rectangle of *width* by *height* pixels.

In other words, this is a convolution. **pbmtopgm** is similar to a special case of **pnmconvol**.

You may need a **ppmsmooth** step after **pbmtopgm**.

pbmtopgm has the effect of anti-aliasing bitmaps which contain distinct line features.

pbmtopgm works best with odd sample width and heights.

You don't need **pbmtopgm** just to use a PGM program on a PBM image. Any PGM program (assuming it uses the Netpbm libraries to read the PGM input) takes PBM input as if it were PGM, with only the mininum and maximum gray levels. So unless your convolution rectangle is bigger than one pixel, you're not gaining anything with a **pbmtopgm** step.

SEE ALSO

netpbm(1), **pgmtopbm**(1), **pbm**(5)

AUTHOR

NAME

pbmtopi3 - convert a portable bitmap into an Atari Degas .pi3 file

SYNOPSIS

pbmtopi3 [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces an Atari Degas .pi3 file as output.

SEE ALSO

pi3topbm(1), pbm(5), ppmtopi1(1), pi1toppm(1)

AUTHOR

Copyright (C) 1988 by David Beckemeyer (bdt!david) and Jef Poskanzer.

NAME

pbmtopk - convert a portable bitmap into a packed (PK) format font

SYNOPSIS

pbmtopk pkfile[.pk] tfmfile[.tfm] resolution [-s designsize] [-p num param...] [-C codingscheme] [-F family] [-f optfile] [-c num] [-W width] [-H height] [-D depth] [-I ital] [-h horiz] [-v vert] [-x xoff] [-y yoff] [pbmfile]...

DESCRIPTION

Reads portable bitmaps as input, and produces a packed (PK) font file and a TFM (TeX font metric) file as output. The resolution parameter indicates the resolution of the font, in dots per inch. If the filename "-" is used for any of the filenames, the standard input stream (or standard output where appropriate) will be used.

OPTIONS

-s designsize
: Sets the design size of the font, in TeX's points (72.27pt to the inch). The default design size is 1. The TFM parameters are given as multiples of the design size.

-p num param...
: Sets the first num font parameters for the font. The first seven parameters are the slant, interword spacing, interword space stretchability, interword space shrinkability, x-height, quad width, and post-sentence extra space of the font. Math and symbol fonts may have more parameters; see The TeXbook for a list of these. Reasonable default values are chosen for parameters which are not specified.

-C codingscheme
: Sets the coding scheme comment in the TFM file.

-F family
: Sets the font family comment in the TFM file.

-f optfile
: Reads the file optfile, which should contain a lines of the form:

```
filename xoff yoff horiz vert width height depth ital
```

The pbm files specified by the filename parameters are inserted consecutively in the font with the specified attributes. If any of the attributes are omitted, or replaced with "*", a default value will be calculated from the size of the bitmap. The settings of the -W, -H, -D, -I, -h, -v, -x, and -y options do not affected characters created in this way. The character number can be changed by including a line starting with "=", followed by the new number. Lines beginning with "%" or "#" are ignored.

-c num
: Sets the character number of the next bitmap encountered to num.

-W width
: Sets the TFM width of the next character to width (in design size multiples).

-H height
: Sets the TFM height of the next character to height (in design size multiples).

-D depth
: Sets the TFM depth of the next character to depth (in design size multiples).

-I ital
: Sets the italic correction of the next character to ital (in design size multiples).

-h horiz
: Sets the horizontal escapement of the next character to horiz (in pixels).

-v vert
: Sets the vertical escapement of the next character to vert (in pixels).

-x xoff
: Sets the horizontal offset of the next character to xoff (in pixels).

-y yoff
: Sets the vertical offset of the next character to yoff (in pixels, from the top row).

SEE ALSO

pktopbm(1), pbm(5)

AUTHOR

Adapted from Tom Rokicki's pxtopk by Angus Duggan <ajcd@dcs.ed.ac.uk>.

NAME

pbmtoplot - convert a portable bitmap into a Unix plot(5) file

SYNOPSIS

pbmtoplot [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a Unix *plot* file.

Note that there is no plottopbm tool - this transformation is one-way.

SEE ALSO

pbm(5), plot(5)

AUTHOR

Copyright (C) 1990 by Arthur David Olson.

NAME

pbmtoppa - convert PBM image to HP Printer Performance Architecture (PPA)

SYNOPSIS

pbmtoppa

[*pbm_fi le* [*ppa_fi le*]]

DESCRIPTION

pbmtoppa converts page images in PBM format to Hewlett Packard's PPA (Printer Performance Architecture) format, which is the data stream format expected by some HP "Windows-only" printers including the HP Deskjet 820C series, the HP DeskJet 720 series, and the HP DeskJet 1000 series.

pbm_fi le is the fi le specifi cation of the input fi le or **-** for Standard Input. The default is Standard Input.

The input fi le contains one or more PBM images, with each one being a single page. Each image must have the exact dimensions of a page (at 600 pixels per inch in both directions). Signifi cantly, this is the format the Ghostscript produces.

ppa_fi le is the fi le specifi cation of the output fi le or **-** for Standard Output. The default is Standard Output.

To print Postscript on an HP PPA printer, just use Ghostscript with the **pbmraw** (or **pbm**) device driver.

You can generate a test page for use with this program with **pbmpage**.

You can also set up a printer fi lter so you can submit PBM input directly to your print queue. See the documentation for your print spooler for information on how to do that, or look in hp820install.doc for an example lpd print fi lter for Postscript and text fi les.

Sometimes, **pbmtoppa** generates a fi le which the printer will not print (because **pbmtoppa**'s input is unprintable). When this happens, all three lights blink to signal the error. This is usually because there is material outside of the printer's printable area. To make the fi le print, increase the margins via **pbmtoppa** options or a confi guration fi le. See the CALIBRATION section below.

-v *version*
printer version (720, 820, or 1000)

-x *xoff* vertical offset adjustment in 1"/600

-y *yoff* horizontal offset adjustment in 1"/600

-t *topmarg*
top margin in 1"/600 (default: 150 = 0.25")

-l *leftmarg*
left margin in 1"/600 (default: 150 = 0.25")

-r *rightmarg*
right margin in 1"/600 (default: 150 = 0.25")

-b *botmarg*
bottom margin in 1"/600 (default: 150 = 0.25")

-s *paper*
paper size: **us** or **a4**. Default is **us**.

-f *cfgfi le*
read parameters from confi guration fi le *cfgfi le*

The **-x** and **-y** options accumulate.

The **-v** option resets the horizontal and vertical adjustments to an internal default.

CONFIGURATION FILES

You can use configuration files to specify parameters rather than use invocation options. **pbmtoppa** processes the file /etc/pbmtoppa.conf, if it exists, before processing any options. It then processes each configuration file named by a **-f** option in order, applying the parameters from the configuration file as if they were invocation options used in the place of the **-f** option.

Configuration files have the following format:

```
#Comment
key1 value1
key2 value2
[etc.]
```

Valid *key*s are **version**, **xoffset**, **yoffset**, **topmargin**, **leftmargin**, **rightmargin**, **bottommargin**, **papersize**, or any non-null prefix of these words. Valid values are the same as with the corresponding invocation parameters.

EXAMPLES

Print a test pattern:

```
pbmpage | pbmppa >/dev/lp1
```

Print three pages:

```
cat page1.pbm page2.pbm page3.pbm | pbmppa >/dev/lp1
```

Print the Postscript file myfile.ps:

```
gs -sDEVICE=rawpbm -q -dNOPAUSE -r600 \
  -sOutputFile=- myfile.ps \
| pbmtoppa | lpr
```

CALIBRATION

To be able to print successfully and properly, you need to tell **pbmtoppa** an X and a Y offset appropriate for your printer to use when generating the page. You can specify these offsets with the **-x** and **-y** invocation options or with the **xoff** and **yoff** parameters in a **pbmtoppa** configuration file.

To determine the correct offsets, use the **pbmpage** program.

If while trying to do this calibration, the printer refuses to print a page, but just blinks all three lights, specify large margins (e.g. 600 pixels -- one inch) via **pbmpage** invocation options while doing the calibration.

For example:

```
pbmpage | pbmtoppa >/dev/lp1
or
pbmpage | pbmtoppa | lpr -l
```

(if your printer filter recognizes the ’-l’ (direct output) parameter).

In the test pattern, the grid is marked off in pixel coordinate numbers. Unfortunately, these coordinates are probably cut off before the edge of the paper. You'll have to use a ruler to estimate the pixel coordinate of the left and top edges of the actual sheet of paper (should be within +/- 300, may be negative; there are 600 pixels per inch).

Add these coordinates to the X and Y offsets by either editing the configuration file or using the **-x** and **-y** command-line parameters.

When **pbmtoppa** is properly calibrated, the center mark should be in the center of the paper. Also, the margins should be able to be as small as 1/4 inch without causing the printer to choke with 'blinking lights syndrome'.

REDHAT LINUX INSTALLATION

RedHat users may find the following tip from Panayotis Vryonis <vrypan@hol.gr> helpful. The same should work for the 820 and 1000, but it hasn't been tested. Also, use the pbmraw GSDriver if you have it; it's faster.

Here is a tip to intergrate HP720C support in RedHat's printtool:

Install pbm2ppa. Copy pbm2ppa to /usr/bin.

Edit "printerdb" (in my system it is found in /usr/lib/rhs/rhs-printfilters) and append the following lines:

```
----------------------Cut here-----------------------
StartEntry: DeskJet720C
  GSDriver: pbm
  Description: {HP DeskJet 720C}
  About: { \
        This driver supports the HP DeskJet 720C \
        inkjet printer. \
        It does does not support color printing. \
        IMPORTANT! Insert \
             "- | pbm2ppa -" \
        in the "Extra GS Otions" field.\
       }
  Resolution: {600} {600} {}
EndEntry --------------------------------------------------
```

Now you can add an HP720C printer just like any other, using printtool.

SEE ALSO

pbmpage(1), **pstopnm**(1), **pbm**(5)

pnm2ppa is not part of Netpbm, but does the same things as **pbmtoppa** except it also works with color and has lots more features. See <http://sourceforge.net/project/?group_id=1322>.

The file INSTALL-MORE in the pbmtoppa directory of the Netpbm source code contains detailed instructions on setting up a system to use pbmtoppa to allow convenient printing on HP PPA printers. It was written by Michael Buehlmann.

For information about the PPA protocol and the separately distributed pbm2ppa program from which **pbmtoppa** was derived, see <http://www.httptech.com/ppa>.

AUTHOR

Tim Norman. Copyright (C) 1998. Licensed under GNU Public License

Manual page by Bryan Henderson, May 2000.

NAME

pbmtopsg3 - convert PBM images to Postscript with G3 fax compression

SYNOPSIS

pbmtopsg3 [**--title=***title*] [**--dpi=***dpi*] [*fi lespec*]

DESCRIPTION

Converts the PBM images in the input PBM fi le to pages in a Postscript fi le encoded with G3 fax compression.

If you don’t specify *fi lespec*, the input is from Standard Input.

Remember that you can create a multi-image PBM fi le simply by concatenating single-image PBM fi les, so if each page is in a different fi le, you might do:

cat faxpage* | pbmtopsg3 >fax.ps

OPTIONS

-title The Postscript title value. Default is no title.

-dpi The resolution of the Postscript output. Default is 72 dpi.

SEE ALSO

pnmtops(1), **pstopnm**(1), **gs**(1), **pstopnm**(1), **pbmtolps**(1), **pbmtoepsi**(1), **pbmtog3**(1), **g3topbm**(1), **pbm**(5)

NAME

pbmtoptx - convert a portable bitmap into Printronix printer graphics

SYNOPSIS

pbmtoptx [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a file of Printronix printer graphics as output.

Note that there is no ptxtopbm tool - this transformation is one way.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pbmtowbmp - convert a portable bitmap (pbm) to a wireless bitmap (wbmp) file

SYNOPSIS

pbmtowbmp [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces a wbmp file as output.

LIMITATIONS

Currently only WBMP type 0 is generated. This is the only type specified in the WAP 1.1 specifications.

SEE ALSO

pbm(5), **wbmptopbm**(1), **Wireless Application Environment Specification.**

AUTHOR

Copyright (C) 1999 Terje Sannum <terje@looplab.com>.

NAME

pbmtox10bm - convert a portable bitmap into an X10 bitmap

SYNOPSIS

pbmtox10bm [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces an X10 bitmap as output. This older format is maintained for compatibility.

Note that there is no x10bmtopbm tool, because *xbmtopbm* can read both X11 and X10 bitmaps.

SEE ALSO

pbmtoxbm(1), xbmtopbm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pbmtoxbm - convert a portable bitmap into an X11 bitmap

SYNOPSIS

pbmtoxbm [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces an X11 bitmap as output.

SEE ALSO

pbmtox10bm(1), xbmtopbm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

pgmtoybm - convert a portable bitmap into a Bennet Yee "face" file

SYNOPSIS

pbmtoybm [*pbmfile*]

DESCRIPTION

Reads a portable bitmap as input. Produces as output a file acceptable to the *face* and *xbm* programs by Bennet Yee (bsy+@cs.cmu.edu).

SEE ALSO

ybmtopbm(1), pbm(5), face(1), face(5), xbm(1)

AUTHOR

Copyright (C) 1991 by Jamie Zawinski and Jef Poskanzer.

NAME

pbmtozinc - convert a portable bitmap into a Zinc bitmap

SYNOPSIS

pbmtozinc [*pbmfi le*]

DESCRIPTION

Reads a portable bitmap as input. Produces a bitmap in the format used by the Zinc Interface Library (ZIL) Version 1.0 as output.

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1988 by James Darrell McCauley (jdm5548@diamond.tamu.edu) and Jef Poskanzer.

NAME

pbmupc - create a Universal Product Code bitmap

SYNOPSIS

pbmupc [**-s1**|**-s2**] *type manufac product*

DESCRIPTION

Generates a Universal Product Code symbol. The three arguments are: a one digit product type, a five digit manufacturer code, and a five digit product code. For example, "0 72890 00011" is the code for Heineken.

As presently configured, *pbmupc* produces a bitmap 230 bits wide and 175 bits high. The size can be altered by changing the defines at the beginning of the program, or by running the output through *pnmenlarge* or *pnmscale*.

OPTIONS

The **-s1** and **-s2** flags select the style of UPC to generate. The default, **-s1**, looks more or less like this:

```
 ||||||||||||||||
 ||||||||||||||||
 ||||||||||||||||
 ||||||||||||||||
0||12345||67890||5
```

The other style, **-s2**, puts the product type digit higher up, and doesn't display the checksum digit:

```
 ||||||||||||||||
 ||||||||||||||||
0||||||||||||||||
 ||||||||||||||||
 ||12345||67890||
```

SEE ALSO

pbm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pcdindex - create index image for a photo CD

SYNOPSIS

pcdindex [**-m** *width*] [**-s** *size*] [**-a** *across*] [**-c** *colors*] [**-f** *font*] [**-b**|**-w**] [*pcdfi le*]

DESCRIPTION

This program generates an index image for a photo CD, based on the photo CD overview fi le.

You can achieve a similar result with **hpcdtoppm -Overview** followed by **pnmindex -black** on the generated PPM images.

OPTIONS

-w*width*
Maximum width of the result image (default: 1152).

-s*size* Maximum size of each of the images (default: 192).

-a*across*
Maximum number of images across (default: 6).

-c*colors*
Maximum number of colors, or **n** to mean no quantization

-f *font* Font to be used for annotation (default: internal font).

-b Black background color (default).

-w White background color.

EXAMPLES

```
pcdindex -m 768 -s 96 -f smallfont.pbm overview.pcd > overview.ppm
pcdindex /cdrom/photo_cd/overview.pcd | ppmtojpeg > overview.jpg
```

SEE ALSO

hpcdtoppm(1), **pnmindex**(1), **ppmtojpeg**(1), **ppm**(5)

NAME

pcxtoppm - convert a PCX file into a portable pixmap

SYNOPSIS

pcxtoppm [**-stdpalette**] [**-verbose**] [*pcxfile*]

DESCRIPTION

Reads a PCX file as input. Produces a portable pixmap as output. Supported PCX types are:

Colormapped files with 2-16 colors.
"Packed pixel" format (1, 2 or 4 bits/pixel, 1 plane) or bitplane format (1 bit/pixel, 1-4 planes). The program checks the colormap and uses an internal one if the provided colormap is completely black.

Colormapped files with 256 colors
8 bits/pixel, 1 plane, colormap at the end of the file.

24bit truecolor files
24bit RGB: 8 bits/pixel, 3 planes.

32bit truecolor files
24bit RGB + 8bit intensity: 8 bits/pixel, 4 planes.

OPTIONS

-stdpalette
Enforce the use of the internal colormap for files with 16 colors or less.

SEE ALSO

ppmtopcx(1), **ppm**(5)

AUTHORS

Copyright (C) 1990 by Michael Davidson.
Modified 1994 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)

NAME

pgmbentley - Bentleyize a portable graymap

SYNOPSIS

pgmbentley [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Performs The Bentley Effect, and writes a portable graymap as output.

The Bentley Effect is described in "Beyond Photography" by Holzmann, chapter 4, photo 4. It's a vertical smearing based on brightness.

SEE ALSO

pgmoil(1), ppmrelief(1), pgm(5)

AUTHOR

Copyright (C) 1990 by Wilson Bent (whb@hoh-2.att.com)

NAME

pgmclouds − generate a random picture of clouds

SYNOPSIS

pgmclouds [**−x** *width*] [**−y** *height*]

DESCRIPTION

pgmclouds generates a picture which looks like a cloudy sky. The algorithm first paints the corner pixels, which may have very different gray scales. In subsequent steps, pixels in between are painted by using an average of neighbour pixels plus/minus a random value, which interval is the smaller the nearer the pixels are to each other.

OPTIONS

−x *width*

Generate a picture which is *width* pixels wide. The default is 512.

−y *height*

Generate a picture which is *height* pixels high. The default is 512.

AUTHOR

Michael Haardt, 1998.

HISTORY

The algorithm used is a slightly modified version of the algorithm presented in “Thomas Graf: *Mathematische Wolken*, Amiga-Magazin 10/1987, p. 91–92”.

SEE ALSO

pgm(5)

NAME

pgmcrater - create cratered terrain by fractal forgery

SYNOPSIS

pgmcrater [**-number** *n*] [**-height**|**-ysize** *s*] [**-width**|**-xsize** *s*] [**-gamma** *g*]

All options can be abbreviated to their shortest unique prefix.

DESCRIPTION

pgmcrater creates a PGM image which mimics cratered terrain. The PGM image is created by simulating the impact of a given number of craters with random position and size, then rendering the resulting terrain elevations based on a light source shining from one side of the screen. The size distribution of the craters is based on a power law which results in many more small craters than large ones. The number of craters of a given size varies as the reciprocal of the area as described on pages 31 and 32 of Peitgen and Saupe[1]; cratered bodies in the Solar System are observed to obey this relationship. The formula used to obtain crater radii governed by this law from a uniformly distributed pseudorandom sequence was developed by Rudy Rucker.

High resolution images with large numbers of craters often benefit from being piped through **pnmsmooth**. The averaging performed by this process eliminates some of the jagged pixels and lends a mellow ‘‘telescopic image’’ feel to the overall picture.

pgmcrater simulates only small craters, which are hemispherical in shape (regardless of the incidence angle of the impacting body, as long as the velocity is sufficiently high). Large craters, such as Copernicus and Tycho on the Moon, have a ‘‘walled plain’’ shape with a cross-section more like:

```
       /\                          /\
_____/    \____________/\____________/    \_____
```

Larger craters should really use this profile, including the central peak, and totally obliterate the pre-existing terrain.

OPTIONS

-number *n* Causes *n* craters to be generated. If no **-number** specification is given, 50000 craters will be generated. Don’t expect to see them all! For every large crater there are many, many more tiny ones which tend simply to erode the landscape. In general, the more craters you specify the more realistic the result; ideally you want the entire terrain to have been extensively turned over again and again by cratering. High resolution images containing five to ten million craters are stunning but take quite a while to create.

-height *height*
Sets the height of the generated image to *height* pixels. The default height is 256 pixels.

-width *width*
Sets the width of the generated image to *width* pixels. The default width is 256 pixels.

-xsize *width*
Sets the width of the generated image to *width* pixels. The default width is 256 pixels.

-ysize *height*
Sets the height of the generated image to *height* pixels. The default height is 256 pixels.

-gamma *factor*
The specified *factor* is used to gamma adjust the image in the same manner as performed by **pnmgamma**. The default value is 1.0, which results in a medium contrast image. Values larger than 1 lighten the image and reduce contrast, while values less than 1 darken the image, increasing contrast.

Note that this is separate from the gamma correction that is part of the definition of the PGM format. The image **pnmgamma** generates is a genuine, gamma-corrected PGM image in any case. This option simply changes the contrast and may compensate for a display device that does not correctly render PGM images.

DESIGN NOTES

The **-gamma** option isn't really necessary since you can achieve the same effect by piping the output from **pgmcrater** through **pnmgamma**. However, **pgmcrater** performs an internal gamma map anyway in the process of rendering the elevation array into the PGM format, so there's no additional overhead in allowing an additional gamma adjustment.

Real craters have two distinct morphologies.

SEE ALSO

pgm(5), **pnmgamma**(1), **pnmsmooth**(1)

[1] Peitgen, H.-O., and Saupe, D. eds., The Science Of Fractal Images, New York: Springer Verlag, 1988.

AUTHOR

John Walker
Autodesk SA
Avenue des Champs-Montants 14b
CH-2074 MARIN
Suisse/Schweiz/Svizzera/Svizra/Switzerland
Usenet: kelvin@Autodesk.com
Fax: 038/33 88 15
Voice: 038/33 76 33

PLUGWARE! If you like this kind of stuff, you may also enjoy ‘‘James Gleick's Chaos--The Software’’ for MS-DOS, available for $59.95 from your local software store or directly from Autodesk, Inc., Attn: Science Series, 2320 Marinship Way, Sausalito, CA 94965, USA. Telephone: (800) 688-2344 toll-free or, outside the U.S. (415) 332-2344 Ext 4886. Fax: (415) 289-4718. ‘‘Chaos--The Software’’ includes a more comprehensive fractal forgery generator which creates three-dimensional landscapes as well as clouds and planets, plus five more modules which explore other aspects of Chaos. The user guide of more than 200 pages includes an introduction by James Gleick and detailed explanations by Rudy Rucker of the mathematics and algorithms used by each program.

NAME

pgmedge - edge-detect a portable graymap

SYNOPSIS

pgmedge [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Outlines the edges, and writes a portable graymap as output. Piping the result through **pgmtopbm -threshold** and playing with the threshold value will give a bitmap of the edges.

The edge detection technique used is to take the Pythagorean sum of two Sobel gradient operators at 90 degrees to each other. For more details see "Digital Image Processing" by Gonzalez and Wintz, chapter 7.

The maxval of the output is the same as the maxval of the input The effect is better with larger maxvals, so you may want to increase the maxval of the input by running it through **pnmdepth** first.

SEE ALSO

pgmenhance(1), **pgmtopbm**(1), **pnmdepth**(1), **pgm**(5), **pbm**(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pgmenhance - edge-enhance a portable graymap

SYNOPSIS

pgmenhance [-*N*] [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Enhances the edges, and writes a portable graymap as output.

The edge enhancing technique is taken from Philip R. Thompson's "xim" program, which in turn took it from section 6 of "Digital Halftones by Dot Diffusion", D. E. Knuth, ACM Transaction on Graphics Vol. 6, No. 4, October 1987, which in turn got it from two 1976 papers by J. F. Jarvis et. al.

OPTIONS

The optional -*N* flag should be a digit from 1 to 9. 1 is the lowest level of enhancement, 9 is the highest, The default is 9.

SEE ALSO

pgmedge(1), pgm(5), pbm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pgmfract − generate fractal as PGM picture

SYNOPSIS

pgmfract [**−s** *size*] [**−d** *depth*] [**−p** *xpos***:***ypos***:***distance*]

DESCRIPTION

pgmfract computes a fractal and outputs it to standard output as PGM file. Each x, y pixel of the picture corresponds to the complex number $c = x + yi$ in the formula $z' = z^2 + c$. The first iteration begins with $z = x + iy$, following iterations use the previously calculated value of z'. As soon as the absolute value of z becomes larger than 2, it is known that the iteration will converge against infinity and the pixel is coloured depending on the number of iterations. Some values of c never cause z to become larger than 2, so the iteration depth has to be limited. If this limit is reached, the pixel will be black. The Mandelbrot set is the set of values for c, which never cause z to converge against infinity, so limiting the depth causes a small error. Using a large enough depth minimizes this error.

OPTIONS

−s *size* Change the picture size from the default of 600 pixel in each direction to *size* pixels.

−d *depth*
Change the maximum iteration depth from the default of 600 iterations to *depth* iterations.

−p *xpos***:***ypos***:***distance*
Change the interval of the complex start values from **0.0:0.0:2.0** to the new center *xpos* an *ypos* with an interval of *distance* in each direction.

−j *cr***:***ci* Compute a julia set instead of a mandelbrot set by specifying a fixed value c for iterations instead of a pixel dependent value $x + iy$. Interesting subsections from the mandelbrot set yield interesting julia sets.

EXAMPLES

Beautiful sections are:

```
pgmfract -p -0.5:0.0:1.5 >fractal.pgm
pgmfract -p -0.7660315:0.100861:0.0003 >fractal.pgm
pgmfract -p -1.252758:0.342541:0.007629 >fractal.pgm
pgmfract -p -0.368056:0.645833:0.097222 >fractal.pgm
pgmfract -p -0.17596915:1.08649105:0.0000004 >fractal.pgm
pgmfract -j -0.17596915:1.08649105 -p 0.0:0.0:0.01 >fractal.pgm
pgmfract -d 2000 -p -0.74567846:0.09998153:0.00012307 >fractal.pgm
pgmfract -d 2000 -j -0.74567846:0.09998153 -p 0.0:0.0:0.1 >fractal.pgm
```

AUTHOR

Michael Haardt <michael@moria.de>.

HISTORY

The original version of this program has been written in 6502 Assembler for an Acorn Electron in 1985. From there, it went to Small-C for the Z80 on a PCW8256 in 1986 to ANSI-C and Linux in 1998.

SEE ALSO

pgm(5)

NAME
pgmhist - print a histogram of the values in a portable graymap

SYNOPSIS
pgmhist [*pgmfile*]

DESCRIPTION
Reads a portable graymap as input. Prints a histogram of the gray values.

SEE ALSO
pgmnorm(1), pgm(5), ppmhist(1)

AUTHOR
Copyright (C) 1989 by Jef Poskanzer.

NAME

pgmkernel - generate a convolution kernel

SYNOPSIS

pgmkernel [**−weight** *w*] *width* [*height*]

DESCRIPTION

Generates a portable graymap array of size *width* x *height* (or *width* x *width* if *height* is not specified) to be used as a convolution file by **pnmconvol**. The data in the convolution array K are computed according to the formula:

K(i,j) = 1 / (1 + w * sqrt((i-width/2)ˆ 2 + (j-height/2)ˆ 2))

where *w* is a coefficient specified via the *−weight* flag, and *width* and *height* are the X and Y filter sizes.

The output PGM file is always written out in ASCII format.

OPTIONS

The optional -*weight* flag should be a real number greater than -1. The default value is 6.0.

BUGS

The computation time is proportional to *width* * *height*. This increases rapidly with the increase of the kernel size. A better approach could be using a FFT in these cases.

SEE ALSO

pnmconvol(1), pnmsmooth(1)

AUTHOR

Alberto Accomazzi (alberto@cfa.harvard.edu).

NAME

pgmmountains − generate a random landscape of mountains and sea

SYNOPSIS

pgmmountains

DESCRIPTION

pgmmountains generates a random landscape of mountains and sea on standard output. The algorithm starts with a single triangle, which is divided in four triangles. The corners of these triangles are modified using random numbers. Each of these triangles is again splitted. If corners fall below heigth zero, they will be painted as sea.

AUTHOR

Michael Haardt, 1998.

HISTORY

The algorithm used is a slightly modified version of the algorithm presented in “Frank/Kreuder, Norbert Siepenkötter: *Gebirge aus dem Computer*, Amiga-Magazin 10/1987, p. 83”.

SEE ALSO

pgm(5)

NAME

pgmnoise - create a graymap made up of white noise

SYNOPSIS

pgmnoise *width height*

DESCRIPTION

Creates a portable graymap that is made up of random pixels with gray values in the range of 0 to PGM_MAXMAXVAL (depends on the compilation, either 255 or 65535). The graymap has a size of width * height pixels.

SEE ALSO

pgm(5)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

pgmramp - generate a grayscale ramp

SYNOPSIS

pgmramp -lr|-tb | -rectangle|-ellipse *width height*

DESCRIPTION

Generates a graymap of the specified size containing a black-to-white ramp. These ramps are useful for multiplying with other images, using the *pnmarith* tool.

OPTIONS

-lr A left to right ramp.

-tb A top to bottom ramp.

-rectangle
A rectangular ramp.

-ellipse An elliptical ramp.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmarith(1), pgm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pgmslice - extract one line of pixel values out of a portable graymap

SYNOPSIS

pgmslice -row|**-col** *line* [*pgmfile*]

DESCRIPTION

Extracts one line of pixel values out of a portable graymap and outputs it in a two column ascii format, with the first value being the pixel's position in the line and the second value the pixel's greyscale value. This is useful for making cross sections through, for example, greyscale CCD images.

OPTIONS

-row Extract a row of pixels

-col Extract a column of pixels

SEE ALSO

pgm(5)

AUTHOR

pgmslice was written by Jos Dingjan <jos@tuatha.org> after being unable to find the source code to Marco Beijersbergen's program with the same name.

NAME

pgmtexture - calculate textural features on a portable graymap

SYNOPSIS

pgmtexture [**-d** *d*] [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Calculates textural features based on spatial dependence matrices at 0, 45, 90, and 135 degrees for a distance *d* (default = 1). Textural features include:

(1) Angular Second Moment,
(2) Contrast,
(3) Correlation,
(4) Variance,
(5) Inverse Difference Moment,
(6) Sum Average,
(7) Sum Variance,
(8) Sum Entropy,
(9) Entropy,
(10) Difference Variance,
(11) Difference Entropy,
(12, 13) Information Measures of Correlation, and
(14) Maximal Correlation Coefficient.

Algorithm taken from:
Haralick, R.M., K. Shanmugam, and I. Dinstein. 1973. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybertinetics,* SMC-3(6):610-621.

BUGS

The program can run incredibly slow for large images (larger than 64 x 64) and command line options are limited. The method for finding (14) the maximal correlation coefficient, which requires finding the second largest eigenvalue of a matrix Q, does not always converge.

REFERENCES

IEEE Transactions on Systems, Man, and Cybertinetics, SMC-3(6):610-621.

SEE ALSO

pgm(5), pnmcut(1)

AUTHOR

Copyright (C) 1991 by Texas Agricultural Experiment Station, employer for hire of James Darrell McCauley.

NAME

pgmtofs - convert portable graymap to Usenix FaceSaver(tm) format

SYNOPSIS

pgmtofs [*pgmfi le*]

DESCRIPTION

Reads a portable graymap as input. Produces Usenix FaceSaver(tm) format as output.

FaceSaver is a registered trademark of Metron Computerware Ltd. of Oakland, CA.

SEE ALSO

fstopgm(1), pgm(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pgmtolispm - convert a portable graymap into Lisp Machine format

SYNOPSIS

pgmtolispm [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Produces a Lisp Machine bitmap as output.

This is the file format read by the tv:read-bit-array-file function on TI Explorer and Symbolics lisp machines.

Given a pgm (instead of a pbm) a multi-plane image will be output. This is probably not useful unless you have a color lisp machine.

Multi-plane bitmaps on lisp machines are color; but the lispm image file format does not include a color map, so we must treat it as a graymap instead. This is unfortunate.

SEE ALSO

lispmtopgm(1), pgm(5)

BUGS

Output width is always rounded up to the nearest multiple of 32; this might not always be what you want, but it probably is (arrays which are not modulo 32 cannot be passed to the Lispm BITBLT function, and thus cannot easily be displayed on the screen).

No color.

AUTHOR

Copyright (C) 1991 by Jamie Zawinski and Jef Poskanzer.

NAME

pgmtopbm - convert a portable graymap into a portable bitmap

SYNOPSIS

pgmtopbm [**-fbyd**|**-fs**|**-threshold** |**-hilbert** |**-dither8**|**-d8**|**-cluster3** |**-c3**|**-cluster4**|**-c4** |**-cluster8**|**-c8**] [**-value** *val*] [**-clump** *size*] [*pgmfile*]

DESCRIPTION

Reads a portable graymap as input. Produces a portable bitmap as output.

Note that there is no pbmtopgm converter. Any program that uses the Netpbm libraries to read PGM files, including virtually all programs in the Netpbm package, will read a PBM file automatically as if it were a PGM file.

If you are using a less intelligent program that expects PGM input, use **pnmdepth** to convert the PBM file to PGM. As long as the depth is greater than 1, **pnmdepth** will generate PGM. This less intelligent program quite probably is also not intelligent enough to deal with general maxvals, so you should specify a depth of 255.

OPTIONS

The default quantization method is boustrophedonic Floyd-Steinberg error diffusion (**-fbyd** or **-fs**). Also available are simple thresholding (**-threshold**); Bayer's ordered dither (**-dither8**) with a 16x16 matrix; and three different sizes of 45-degree clustered-dot dither (**-cluster3**, **-cluster4**, **-cluster8**). A space filling curve halftoning method using the Hilbert curve is also available. (**-hilbert**);

Floyd-Steinberg will almost always give the best looking results; however, looking good is not always what you want. For instance, thresholding can be used in a pipeline with the *pnmconvol* tool, for tasks like edge and peak detection. And clustered-dot dithering gives a newspaper-ish look, a useful special effect.

The **-value** flag alters the thresholding value for Floyd-Steinberg and simple thresholding. It should be a real number between 0 and 1. Above 0.5 means darker images; below 0.5 means lighter.

The Hilbert curve method is useful for processing images before display on devices that do not render individual pixels distinctly (like laser printers). This dithering method can give better results than the dithering usually done by the laser printers themselves. The **-clump** flag alters the number of pixels in a clump. This is usually an integer between 2 and 100 (default 5). Smaller clump sizes smear the image less and are less grainy, but seem to loose some grey scale linearity. Typically a PGM image will have to be scaled to fit on a laser printer page (2400 x 3000 pixels for an A4 300 dpi page), and then dithered to a PBM image before being converted to a postscript file. A printing pipeline might look something like: pnmscale -xysize 2400 3000 image.pgm | pgmtopbm -hil | pnmtops -scale 0.25 > image.ps

All flags can be abbreviated to their shortest unique prefix.

REFERENCES

The only reference you need for this stuff is "Digital Halftoning" by Robert Ulichney, MIT Press, ISBN 0-262-21009-6.

The Hilbert curve space filling method is taken from "Digital Halftoning with Space Filling Curves" by Luiz Velho, Computer Graphics Volume 25, Number 4, proceedings of SIGRAPH '91, page 81. ISBN 0-89791-436-8

SEE ALSO

pbmreduce(1), pgm(5), pbm(5), pnmconvol(1), pnmscale(1), pnmtops(1)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pgmtoppm - colorize a PGM (grayscale) image into a PGM (color) image

SYNOPSIS

pgmtoppm *colorspec* [*pgmfile*]
pgmtoppm *colorspec1***-***colorspec2* [*pgmfile*]
pgmtoppm -map *mapfile* [*pgmfile*]

DESCRIPTION

Reads a PGM as input. Produces a PPM file as output with a specific color assigned to each gray value in the input.

If you specify one color argument, black in the pgm file stays black and white in the pgm file turns into the specified color in the ppm file. Gray values in between are linearly mapped to differing intensities of the specified color.

If you specify two color arguments (separated by a dash), then black gets mapped to the first color and white gets mapped to the second and gray values in between get mapped linearly (across a three dimensional space) to colors in between.

You can specify the color in one of five ways:

- o A name, from an X11-style color names file.
- o An X11-style hexadecimal specifier: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.
- o An X11-style decimal specifier: rgbi:r/g/b, where r g and b are floating point numbers between 0 and 1.
- o For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.
- o For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are floating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

Also, you can specify an entire colormap with the **-map** option. The mapfile is just a **ppm** file; it can be any shape, all that matters is the colors in it and their order. In this case, black gets mapped into the first color in the map file, and white gets mapped to the last and gray values in between are mapped linearly onto the sequence of colors in between.

NOTE - MAXVAL

The "maxval," or depth, of the output image is the same as that of the input image. The maxval affects the color resolution, which may cause quantization errors you don't anticipate in your output. For example, you have a simple black and white image (in fact, let's say it's a PBM file, since **pgmtoppm**, like all Netpbm programs, can accept a PBM file as if it were PGM. The maxval of this image is 1, because only two gray values are needed: black and white. Run this image through **pgmtoppm 0f/00/00** to try to make the image black and faint red. Because the output image will also have maxval 1, there is no such thing as faint red. It has to be either full-on red or black. **pgmtoppm** rounds the color 0f/00/00 down to black, and you get an output image that is nothing but black.

The fix is easy: Pass the input through **pnmdepth** on the way into **pgmtoppm** to increase its depth to something that would give you the resolution you need to get your desired color. In this case, **pnmdepth 16** would do it. Or spare yourself the unnecessary thinking and just say **pnmdepth 255 .**

SEE ALSO

pnmdepth(1), **rgb3toppm**(1), **ppmtopgm**(1), **ppmtorgb3**(1), **ppm**(5), **pgm(5)**

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pi1toppm - convert an Atari Degas .pi1 into a portable pixmap

SYNOPSIS

pi1toppm [*pi1fi le*]

DESCRIPTION

Reads an Atari Degas .pi1 fi le as input. Produces a portable pixmap as output.

SEE ALSO

ppmtopi1(1), ppm(5), pi3topbm(1), pbmtopi3(1)

AUTHOR

Copyright (C) 1991 by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

pi3topbm - convert an Atari Degas .pi3 file into a portable bitmap

SYNOPSIS

pi3topbm [*pi3file*]

DESCRIPTION

Reads an Atari Degas .pi3 file as input. Produces a portable bitmap as output.

SEE ALSO

pbmtopi3(1), pbm(5), pi1toppm(1), ppmtopi1(1)

AUTHOR

Copyright (C) 1988 by David Beckemeyer (bdt!david) and Diomidis D. Spinellis.

NAME

picttoppm - convert a Macintosh PICT file into a portable pixmap

SYNOPSIS

picttoppm [**-verbose**] [**-fullres**] [**-noheader**] [**-quickdraw**] [**-fontdir**file**]** [*pictfile*]

DESCRIPTION

Reads a PICT file (version 1 or 2) and outputs a portable pixmap. Useful as the first step in converting a scanned image to something that can be displayed on Unix.

OPTIONS

−fontdir *file*
: Make the list of BDF fonts in ‘‘file’’ available for use by *picttoppm* when drawing text. See below for the format of the fontdir file.

−fullres
: Force any images in the PICT file to be output with at least their full resolution. A PICT file may indicate that a contained image is to be scaled down before output. This option forces images to retain their sizes and prevent information loss. Use of this option disables all PICT operations except images.

−noheader
: Do not skip the 512 byte header that is present on all PICT files. This is useful when you have PICT data that was not stored in the data fork of a PICT file.

−quickdraw
: Execute only pure quickdraw operations. In particular, turn off the interpretation of special PostScript printer operations.

−verbose
: Turns on verbose mode which prints a a whole bunch of information that only *picttoppm* hackers really care about.

BUGS

The PICT file format is a general drawing format. *picttoppm* does not support all the drawing commands, but it does have full support for any image commands and reasonable support for line, rectangle, polgon and text drawing. It is useful for converting scanned images and some drawing conversion.

Memory is used very liberally with at least 6 bytes needed for every pixel. Large bitmap PICT files will likely run your computer out of memory.

FONT DIR FILE FORMAT

picttoppm has a built in default font and your local installer probably provided adequate extra fonts. You can point *picttoppm* at more fonts which you specify in a font directory file. Each line in the file is either a comment line which must begin with ‘‘#’’ or font information. The font information consists of 4 whitespace spearated fields. The first is the font number, the second is the font size in pixels, the third is the font style and the fourth is the name of a BDF file containing the font. The BDF format is defined by the X window system and is not described here.

The font number indicates the type face. Here is a list of known font numbers and their faces.

0	Chicago
1	application font
2	New York
3	Geneva
4	Monaco
5	Venice
6	London
7	Athens
8	San Franciso
9	Toronto
11	Cairo
12	Los Angeles
20	Times Roman
21	Helvetica

22	Courier
23	Symbol
24	Taliesin

The font style indicates a variation on the font. Multiple variations may apply to a font and the font style is the sum of the variation numbers which are:

1	Boldface
2	Italic
4	Underlined
8	Outlined
16	Shadow
32	Condensed
64	Extended

Obviously the font defińtions are strongly related to the Macintosh. More font numbers and information about fonts can be found in Macintosh documentation.

SEE ALSO

Inside Macintosh volumes 1 and 5, ppmtopict(1), ppm(5)

AUTHOR

Copyright 1993 George Phillips

NAME

pjtoppm - convert an HP PaintJet file to a portable pixmap

SYNOPSIS

pjtoppm [*paintjet*]

DESCRIPTION

Reads an HP PaintJet file as input and converts it into a portable pixmap. This was a quick hack to save some trees, and it only handles a small subset of the paintjet commands. In particular, it will only handle enough commands to convert most raster image files.

REFERENCES

HP PaintJet XL Color Graphics Printer User's Guide

SEE ALSO

ppmtopj(1)

AUTHOR

Copyright (C) 1991 by Christos Zoulas.

NAME

pktopbm - convert packed (PK) format font into portable bitmap(s)

SYNOPSIS

pktopbm pkfile[.pk] [-x width] [-y height] [-c num] pbmfile ...

DESCRIPTION

Reads a packed (PK) font file as input, and produces portable bitmaps as output. If the filename "-" is used for any of the filenames, the standard input stream (or standard output where appropriate) will be used. If either the width or height is specified, this value will be used for all bitmaps produced. Also if one or both values are specified, the bitmap will be relocated with the hoffset and voffset given in the pkfile. The basepoint will be placed in the lower left corner of the bitmap if the bitmap is bigger than the specified size it will be truncated at the top or right.

OPTIONS

-c num Sets the character number of the next bitmap written to num.

-x width
Sets the width of the bitmap.

-y width
Sets the height of the bitmap.

SEE ALSO

pbmtopk(1), pbm(5)

AUTHOR

Adapted from Tom Rokicki's pxtopk by Angus Duggan <ajcd@dcs.ed.ac.uk>. <bartel@informatik.tu-muenchen.de> in March 1995.

NAME

pngtopnm - convert a Portable Network Graphics file into a portable anymap

SYNOPSIS

pngtopnm [-verbose] [-alpha | -mix] [-background color]
[-gamma value] [-text file] [-time] [*pngfile*]

DESCRIPTION

Reads a Portable Network Graphics as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's black & white, a *pbm* file is written, else if it's grayscale a *pgm* file, else a *ppm* file.

OPTIONS

-verbose
Display the format of the input file and the type of the output file. If the chunks are part of the *png-file,* the alpha, transparency and gamma-values will be indicated.

-alpha Output the alpha channel or transparency mask of the image. The result is either a *pbm* file or *pgm* file, depending on whether different levels of transparency appear.

-mix Compose the image with the transparency or alpha mask against a the background. When a background chunk is available that color is taken, else black will do.

-background color
If no background color chunck is present in the *png-file,* or when another color is required this parameter can be used to set the background color of images. This is especially useful for alpha-channel images or those with transparency chunks. The format, to specify the color in, is either (in the case of orange) "1.0,0.5,0.0", where the values are floats between zero and one, or with the syntax "#RGB", "#RRGGBB" or "#RRRRGGGGBBBB" where R, G and B are hexa-decimal numbers.

-gamma value
Converts the image to a new display-gamma value. When a gAMA chunk is present in the *png-file,* the image-gamma value will be used. When not, the image-gamma is considered to be 1.0. Based on the image-gamma and the display-gamma given with this option the colors written to the *pnm-file* will be adjusted.
Because the gamma's of uncompensated monitors are around 2.6, which results in an image-gamma of 0.45, some typical situations are: when the image-gamma is 0.45 (use -verbose to check) and the picture is too light, your system is gamma-corrected, so convert with "-gamma 1.0". When no gAMA chunk is present or the image-gamma is 1.0, use 2.2 to make the picture lighter and 0.45 to make the picture darker.

-text file
Writes the tEXt and zTXt chunks to a file, in a format as described in the *pnmtopng* man-page. These chunks contain text comments or annotations.

-time Prints the tIME chunk to stderr.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmtopng(1), ptot(1), pnmgamma(1), pnm(5)

NOTE

Instead of pngtopnm|pnmtoxxx, a specific converter should be used, if available. E.g. *ptot* (PNG to TIFF conversion), etc.

BUGS

There could be an option to read the comment text from pnm comments instead of a separate file.

The program could be much faster, with a bit of code optimizing.

AUTHORS

Copyright (C) 1995-1997 by Alexander Lehmann
and Willem van Schaik.

NAME

pnmalias - antialias a portable anyumap.

SYNOPSIS

pnmalias [**-bgcolor** *color*] [**-fgcolor** *color*] [**-bonly**] [**-fonly**] [**-balias**] [**-falias**] [**-weight** *w*] [*pnmfi le*]

DESCRIPTION

Reads a portable anymap as input, and applies anti-aliasing to background and foreground pixels. If the input fi le is a portable bitmap, the output anti-aliased image is promoted to a graymap, and a message is printed informing the user of the change in format.

OPTIONS

−bgcolor *colorb,* **−fgcolor** *colorf*

set the background color to *colorb,* and the foreground to color to *colorf.* Pixels with these values will be anti-aliased. by default, the background color is taken to be black, and foreground color is assumed to be white. The colors can be specifi ed in fi ve ways:

- **o** A name, assuming that a pointer to an X11-style color names fi le was compiled in.
- **o** An X11-style hexadecimal specifi er: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.
- **o** An X11-style decimal specifi er: rgbi:r/g/b, where r g and b are fbating point numbers between 0 and 1.
- **o** For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.
- **o** For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are fbating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

Note that even when dealing with graymaps, background and foreground colors need to be specifi ed in the fashion described above. In this case, background and foreground pixel values are taken to be the value of the red component for the given color.

−bonly, **−fonly**

Apply anti-aliasing only to background (**−bonly**), or foreground (**−fonly**) pixels.

−balias, **−falias**

Apply anti-aliasing to all pixels surrounding background (**−balias**), or foreground (**−falias**) pixels. By default, anti-aliasing takes place only among neighboring background and foreground pixels.

−weight *w*

Use *w* as the central weight for the aliasing fi lter. *W* must be a real number in the range 0 < *w* < 1. The lower the value of *w* is, the "blurrier" the output image is. The default is w = 1/3.

SEE ALSO

pbmtext(1), pnmsmooth(1), pnm(5)

AUTHOR

Copyright (C) 1992 by Alberto Accomazzi, Smithsonian Astrophysical Observatory.

NAME

pnmarith - perform arithmetic on two portable anymaps

SYNOPSIS

pnmarith -add|-subtract|-multiply|-difference|-minimum|-maximum. *pnmfi le1 pnmfi le2*

DESCRIPTION

Reads two portable anymaps as input. Performs the specifi ed arithmetic operation, and produces a portable anymap as output. The two input anymaps must be the same width and height.

The arithmetic is performed between corresponding pixels in the two anymaps, as if maxval was 1.0, black was 0.0, and a linear scale in between. Results that fall outside of [0..1) are truncated.

The operator *-difference* calculates the absolute value of *pnmarith -subtract pnmfi le1 pnmfi le2,* i.e. no truncation is done.

All fl ags can be abbreviated to their shortest unique prefi x.

SEE ALSO

pbmmask(1), pnmpaste(1), pnminvert(1), pnm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer. Lightly modifi ed by Marcel Wijkstra <wijkstra@fwi.uva.nl>

NAME

pnmcat - concatenate portable anymaps

SYNOPSIS

pnmcat [**-white**|**-black**] **-leftright**|**-lr** [**-jtop**|**-jbottom**] *pnmfi le pnmfi le* ...
pnmcat [**-white**|**-black**] **-topbottom**|**-tb** [**-jleft**|**-jright**] *pnmfi le pnmfi le* ...

DESCRIPTION

Reads portable anymaps as input. Concatenates them either left to right or top to bottom, and produces a portable anymap as output.

pamdice splits an image up into smaller ones.

pnmtile concatenates a single input image to itself repeatedly.

OPTIONS

If the anymaps are not all the same height (left-right) or width (top-bottom), the smaller ones have to be justifi ed with the largest. By default, they get centered, but you can specify one side or the other with one of the -j* fhgs. So, **-topbottom -jleft** would stack the anymaps on top of each other, flush with the left edge.

The **-white** and **-black** fhgs specify what color to use to fi ll in the extra space when doing this justifi cation. If neither is specifi ed, the program makes a guess.

All fhgs can be abbreviated to their shortest unique prefi x.

SEE ALSO

pamdice(1), **pnmtile**(1), **pamcut**(1), **pnm**(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmcolormap - create quantization color map for a Netpbm image

SYNOPSIS

pnmcolormap [**-center**|**-meancolor**|**-meanpixel**] [**-spreadbrightness**|**-spreadluminosity**] [**-sort**] [**-square**] *ncolors*|**all** [*pnmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or an equals sign between an option name and its value.

DESCRIPTION

Reads a PNM image as input. Chooses *ncolors* colors to best represent the image, maps the existing colors to the new ones, and writes a PNM color map defi ning them as output.

You can use this map as input to **pnmremap** on the same input image to quantize the colors in that image, I.e. produce a similar image with fewer colors. **pnmquant** does both the **pnmcolormap** and **pnmremap** steps for you.

A PNM colormap is a PNM image of any dimensions that contains at least one pixel of each color in the set of colors it represents.

The quantization method is Heckbert's "median cut". See the section QUANTIZATION METHOD.

If the input image is a PPM, the output image is a PPM. If the input image is a PBM or PGM, the output colormap is a PGM. Note that a colormap of a PBM image is not very interesting.

The colormap generally has the same maxval as the input image, but **pnmcolormap** may reduce it if there are too many colors in the input, as part of its quantization algorithm.

If you want to create a colormap without basing it on the colors in an input image, see **ppmcolors**.

PARAMETERS

The single parameter, which is required, is the number of colors you want in the output colormap. **pnmcolormap** may produce a color map with slightly fewer colors than that. You may specify **all** to get a colormap of every color in the input image (no quantization).

OPTIONS

-sort This option causes the output colormap to be sorted by the red component intensity, then the green, then the blue in ascending order. This is an insertion sort, so it is not very fast on large colormaps. Sorting is useful because it allows you to compare two sets of colors.

-square
By default, **pnmcolormap** produces as the color map a PPM image with one row and one column for each color in the colormap. This option causes **pnmcolormap** instead to produce a PPM image that is within one row or column of being square, with multiple pixels of the same color as necessary to create a number of pixels which is a perfect square.

-verbose
This option causes **pnmcolormap** to display messages to Standard Error about the quantization.

-center

-meancolor

-meanpixel

-spreadbrightness

-spreadluminosity
These options control the quantization algorithm. See QUANTIZATION METHOD below.

QUANTIZATION METHOD

A quantization method is a way to choose which colors, being fewer in number than in the input, you want in the output. **pnmcolormap** uses Heckbert's "median cut" quantization method.

This method involves separating all the colors into "boxes," each holding colors that represent about the same number of pixels. You start with one box and split boxes in two until the number of boxes is the same as the number of colors you want in the output, and choose one color to represent each box.

When you split a box, you do it so that all the colors in one sub-box are "greater" than all the colors in the other. "Greater," for a particular box, means it is brighter in the color component (red, green, blue) which has the largest spread in that box. **pnmcolormap** gives you two ways to defi ne "largest spread.": 1) largest spread of brightness; 2) largest spread of contribution to the luminosity of the color. E.g. red is weighted much more than blue. Select among these with the **-spreadbrightness** and **-spreadluminosity** options. The default is **-spreadbrightness**.

pnmcut provides three ways of choosing a color to represent a box: 1) the center color - the color halfway between the greatest and least colors in the box, using the above defi nition of "greater"; 2) the mean of the colors (each component averaged separately by brightness) in the box; 3) the mean weighted by the number of pixels of a color in the image.

Note that in all three methods, there may be colors in the output which do not appear in the input at all.

Select among these with the **-center**, **-meancolor**, and **-meanpixel** options. The default is **-center**.

REFERENCES

"Color Image Quantization for Frame Buffer Display" by Paul Heckbert, SIGGRAPH '82 Proceedings, page 297.

SEE ALSO

pnmremap(1), **pnmquant**(1), **ppmquantall**(1), **pnmdepth**(1), **ppmdither**(1), **ppmquant**(1), **ppm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer. Copyright (C) 2001 by Bryan Henderson.

NAME

pnmcomp - composite (overlay) two portable anymap files together

SYNOPSIS

pnmcomp [**-xoff=***X* | **-align=**{**left**,**center**,**right**}]
[**-yoff=***Y* | **-valign=**{**top**,**middle**,**bottom**}]
[**-alpha=***alpha-pgmfile*] [**-invert**]
overlay [*pnm-input*] [*pnm-output*]

Minimum unique abbreviations are acceptable.

DESCRIPTION

pnmcomp reads two images and produces a composite image with one of the images overlayed on top of the other. The images need not be the same size. The input and outputs are PNM format image files.

In its simplest use, **pnmcomp** simply places the *overlay* file on top of the *pnm-input* file, blocking out the part of the *pnm-input* file beneath it. If you specify the *alpha-pgmfile*, **pnmcomp** uses it as an alpha mask, which means it determines the level of transparency of each point in the overlay image. The alpha mask must have the same dimensions as the overlay image. In places where the alpha mask defines the overlay image to be opaque, the composite output contains only the contents of the overlay image; the underlying image is totally blocked out. In places where the alpha mask defines the overlay image to be transparent, the composite output contains none of the overlay image; the underlying image shows through completely. In places where the alpha mask shows a value in between opaque and transparent (translucence), the composite image contains a mixture of the overlay image and the underlying image and the level of translucence determines how much of each.

The alpha mask is a PGM file in which a white pixel represents opaqueness and a black pixel transparency. Anything in between is translucent.

In some image file formats (PNG, for example), transparency information (the alpha mask) is part of the definition of the image. In the PNM formats, transparency is always embodied in a separate companion file. The PNM converter programs that convert from an image format such as PNG have options that allow you to extract the transparency information to a separate file, which you can then use as input to **pnmcomp**.

The output image is always of the same dimensions as the underlying image. **pnmcomp** only uses parts of the overlay image that fit within the underlying image.

To specify where on the underlying image to place the overlay image, use the **-xoff**, **-yoff**, **-align**, and **-valign** options. Without these options, the default horizontal position is flush left and the default vertical position is flush top.

The overlay and underlying images may be of different formats (e.g. overlaying a PBM text image over a full color PPM image) and have different maxvals. The output image has the more general of the two input formats and a maxval that is the least common multiple the two maxvals (or the maximum maxval allowable by the format, if the LCM is more than that).

OPTIONS

-invert This option inverts the sense of the values in the alpha mask, which effectively switches the roles of the overlay image and the underlying image in places where the two intersect.

-xoff *X*

-yoff *Y* These options position the overlay image with respect to the underlying image. *X* and *Y* are the horizontal and vertical displacements of the top left corner of the overlay image from the top left corner of the underlying image, in pixels. A positive value means right or down; a negative value means left or up. The overlay need not fit entirely (or at all) on the underlying image. **pnmcomp** uses only the parts that lie over the underlying image.

-align=[**left**,**center**,**right**]
This option is an alternative to **-xoff**, in the style of HTML. It selects the horizontal position of the overlay image so that it is flush left, centered, or flush right on the underlying image.

-valign=[**top**,**middle**,**bottom**]
This option is an alternative to **-yoff**, in the style of HTML. It selects the vertical position of the overlay image so that it is flush top, centered, or flush bottom on the underlying image.

SEE ALSO

ppmmix(1) and **pnmpaste**(1) are simpler, less general versions of the same tool.

pnm(5), **pbmmask**(1)

AUTHOR

Copyright (C) 1992 by David Koblas (koblas@mips.com).

NAME

pnmconvol - general MxN convolution on a portable anymap

SYNOPSIS

pnmconvol *convolutionfile* [*pnmfile*]

DESCRIPTION

Reads two portable anymaps as input. Convolves the second using the first, and writes a portable anymap as output.

Convolution means replacing each pixel with a weighted average of the nearby pixels. The weights and the area to average are determined by the convolution matrix. The unsigned numbers in the convolution file are offset by -maxval/2 to make signed numbers, and then normalized, so the actual values in the convolution file are only relative.

Here is a sample convolution file; it does a simple average of the nine immediate neighbors, resulting in a smoothed image:

```
P2
3 3
18
10 10 10
10 10 10
10 10 10
```

To see how this works, do the above-mentioned offset: 10 - 18/2 gives 1. The possible range of values is from 0 to 18, and after the offset that's -9 to 9. The normalization step makes the range -1 to 1, and the values get scaled correspondingly so they become 1/9 - exactly what you want. The equivalent matrix for 5x5 smoothing would have maxval 50 and be filled with 26.

The convolution file will usually be a graymap, so that the same convolution gets applied to each color component. However, if you want to use a pixmap and do a different convolution to different colors, you can certainly do that.

At the edges of the convolved image, where the convolution matrix would extend over the edge of the image, **pnmconvol** just copies the input pixels directly to the output.

SEE ALSO

pnmsmooth(1), **pnm**(5)

AUTHORS

Copyright (C) 1989, 1991 by Jef Poskanzer.
Modified 26 November 1994 by Mike Burns, burns@chem.psu.edu

NAME

pnmcrop - crop a portable anymap

SYNOPSIS

pnmcrop [**-white**|**-black**|**-sides**] [**-left**] [**-right**] [**-top**] [**-bottom**] [*pnmfi le*]

All options may be abbreviated to their shortest unique prefi x or specifi ed with double hyphens.

DESCRIPTION

Reads a PBM, PGM, or PPM image as input. Removes borders that are the background color, and produces the same type of image as output.

If you don't specify otherwise, **pnmcrop** assumes the background color is whatever color the top left and right corners of the image are and if they are different colors, something midway between them. You can specify that the background is white or black with the **-white** and **-black** options or make **pnmcrop** base its guess on all four corners instead of just two with **-sides**.

By default, **pnmcrop** chops off any stripe of background color it fi nds, on all four sides. You can tell **pnmcrop** to remove only specifi c borders with the **-left**, **-right**, **-top**, and **-bottom** options.

If you want to chop a specifi c amount off the side of an image, use **pnmcut**.

If you want to add different borders after removing the existing ones, use **pnmcat** or **pnmcomp**.

OPTIONS

-white Take white to be the background color. **pnmcrop** removes borders which are white.

-black Take black to be the background color. **pnmcrop** removes borders which are black.

-sides Determine the background color from the colors of the four corners of the input image. **pnmcrop** removes borders which are of the background color.

If at least three of the four corners are the same color, **pnmcrop** takes that as the background color. If not, **pnmcrop** looks for two corners of the same color in the following order, taking the fi rst found as the background color: top, left, right, bottom. If all four corners are different colors, **pnmcrop** assumes an average of the four colors as the background color.

The **-sides** option slows **pnmcrop** down, as it reads the entire image to determine the background color in addition to the up to three times that it would read it without **-sides**.

-left Remove any left border.

-right Remove any right border.

-top Remove any top border.

-bottom
Remove any bottom border.

-verbose
Print on Standard Error information about the processing, including exactly how much is being cropped off of which sides.

SEE ALSO

pnmcut(1), **pnmfi le**(1), **pnm**(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmcut - cut a rectangle out of a PBM, PGM, or PPM image

SYNOPSIS

pnmcut [**-left** *leftcol*] [**-right** *rightcol*] [**-top** *toprow*] [**-bottom** *bottomrow*] [**-width** *width*] [**-height** *height*] [**-pad**] [**-verbose**] [*left top width height*] [*pnmfi le*]

All options may be abbreviated to the shortest unique prefi x.

DESCRIPTION

Reads a PBM, PGM, or PPM image as input. Extracts the specifi ed rectangle, and produces the same kind of image as output.

There are two ways to specify the rectangle to cut: arguments and options. Options are easier to remember and read, more expressive, and allow you to use defaults. Arguments were the only way available before July 2000.

If you use both options and arguments, the two specifi cations get mixed in an unspecifi ed way.

To use options, just code any mixture of the **-left**, **-right**, **-top**, **-bottom**, **-width**, and **-height** options. What you don't specify defaults. It is an error to overspecify, i.e. to specify all three of **-left**, **-right**, and **-width** or **-top**, **-bottom**, and **-height**.

To use arguments, specify all four of the *left*, *top*, *width*, and *height* arguments. *left* and *top* have the same effect as specifying them as the argument of a **-left** or **-top** option, respectively. *width* and *height* have the same effect as specifying them as the argument of a **-width** or **-height** option, respectively, where they are positive. Where they are not positive, they have the same effect as specifying one less than the value as the argument to a **-right** or **-bottom** option, respectively. (E.g. *width* = 0 makes the cut go all the way to the right edge). Before July 2000, negative numbers were not allowed for *width* and *height*.

Input is from Standard Input if you don't specify the input fi le *pnmfi le*.

Output is to Standard Output.

OPTIONS

-left The column number of the leftmost column to be in the output. If a nonnegative number, it refers to columns numbered from 0 at the left, increasing to the right. If negative, it refers to columns numbered -1 at the right, decreasing to the left.

-right The column number of the rightmost column to be in the output, numbered the same as for **-left.**

-top The row number of the topmost row to be in the output. If a nonnegative number it refers to rows numbered from 0 at the top, increasing downward. If negative, it refers to columns numbered -1 at the bottom, decreasing upward.

-bottom
The row number of the bottom-most row to be in the output, numbered the same as for **-top**.

-width The number of columns to be in the output. Must be positive.

-height The number of rows to be in the output. Must be positive.

-pad If the rectangle you specify is not entirely within the input image, **pnmcut** fails unless you also specify **-pad**. In that case, it pads the output with black up to the edges you specify. You can use this option if you need to have an image of certain dimensions and have an image of arbitrary dimensions.

pnmpad can also fi ll an image out to a specifi ed dimension, and gives you more explicit control over the padding.

-verbose
Print information about the processing to Standard Error.

SEE ALSO

pnmcrop(1), **pnmpad**(1), **pnmcat**(1), **pgmslice**(1), **pnm**(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmdepth - change the maxval in a portable anymap

SYNOPSIS

pnmdepth *newmaxval* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Scales all the pixel values, and writes out the image with the new maxval. Scaling the colors down to a smaller maxval will result in some loss of information.

Be careful of off-by-one errors when choosing the new maxval. For instance, if you want the color values to be five bits wide, use a maxval of 31, not 32.

One important use of **pnmdepth** is to convert a new format 2-byte-per-sample PNM file to the older 1-byte-per-sample format. Before April 2000, essentially all raw (binary) format PNM files had a maxval less than 256 and one byte per sample, and many programs may rely on that. If you specify a *newmaxval* less than 256, the resulting file should be readable by any program that worked with PNM files before April 2000.

SEE ALSO

pnm(5), ppmquant(1), ppmdither(1)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmenlarge - read a portable anymap and enlarge it N times

SYNOPSIS

pnmenlarge *N* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Replicates its pixels *N* times, and produces a portable anymap as output.

pnmenlarge can only enlarge by integer factors. The slower but more general *pnmscale* can enlarge or reduce by arbitrary factors, and *pbmreduce* can reduce by integer factors, but only for bitmaps.

If you enlarge by a factor of 3 or more, you should probably add a *pnmsmooth* step; otherwise, you can see the original pixels in the resulting image.

SEE ALSO

pbmreduce(1), pnmscale(1), pnmsmooth(1), pnm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmfile - describe a portable anymap

SYNOPSIS

pnmfile [**-allimages**] [*pnmfile* ...]

DESCRIPTION

Reads one or more Netpbm files as input. Writes out short descriptions of the image type, size, etc. This is mostly for use in shell scripts, so the format is not particularly pretty.

OPTIONS

-allimages

Describe every image in each input file. Without this option, **pnmfile** describes only the first image in each input file. Note that before July 2000, a file could not contain more than one image and many programs ignore all but the first.

SEE ALSO

pnm(5), **file**(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pnmflip - perform one or more flip operations on a portable anymap

SYNOPSIS

pnmflip [**-leftright**|**-lr**] [**-topbottom**|**-tb**] [**-transpose**|**-xy**] [**-rotate90**|**-r90**|**-ccw**] [**-rotate270**|**-r270**|**-cw**] [**-rotate180**|**-r180**] [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Performs one or more flip operations, in the order specified, and writes out a portable anymap.

OPTIONS

The flip operations available are: left for right (**-leftright** or **-lr**); top for bottom (**-topbottom** or **-tb**); and transposition (**-transpose** or **-xy**). In addition, some canned concatenations are available: **-rotate90** or **-ccw** is equivalent to **-transpose -topbottom**; **-rotate270** or **-cw** is equivalent to **-transpose -leftright**; and **-rotate180** is equivalent to **-leftright -topbottom**.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmrotate(1), pnm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmgamma - perform gamma correction on a PNM image

SYNOPSIS

pnmgamma [**-ungamma**] [**-cieramp**|**-srgbramp**] [*value* [*pnmfi le*]]
pnmgamma [**-ungamma**] [**-cieramp**|**-srgbramp**] *redgamma greengamma bluegamma* [*pnmfi le*]

DESCRIPTION

Performs gamma correction on pseudo-PNM images.

The PPM format specifi cation specify that certain sample values in a fi le represent certain light intensities in an image. In particular, they specify that the sample values are directly proportional to gamma-corrected intensity values. The gamma correction they specify is CIE Rec. 709.

However, people sometimes work with approximations of PPM and PGM where the relationship between the image intensities and the sample values are something else. For example, the sample value might be directly proportional to the intensity with no gamma correction (often called "linear intensity"). Or a different gamma transfer function may be used.

pnmgamma allows you to manipulate the transfer function, thus working with and/or creating pseudo-PPM fi les that are useful for various things.

For example, if you feed a true PPM to **pnmgamma -cieramp -ungamma**, you get as output a fi le which is PPM in every respect except that the sample values are directly proportional to the light intensities in the image. If you feed such a fi le to **pnmgamma -cieramp**, you get out a true PPM.

The situation for PGM images is analogous. And **pnmgamma** treats PBM images as PGM images.

When you feed a linear PPM image to a display program that expects a true PPM, the display appears darker than it should, so **pnmgamma** has the effect of lightening the image. When you feed a true PPM to a display program that expects linear sample values, and therefore does a gamma correction of its own on them, the display appears lighter than it should, so **pnmgamma** with a gamma value less than one (the multiplicative inverse of whatever gamma value the display program uses) has the effect of darkening the image.

PARAMETERS

The only parameters are the specifi cation of the input image fi le and the gamma values. Every gamma transfer function **pnmgamma** uses contains an exponent, which is the gamma value, and you can choose that value.

Furthermore, you can choose different values for each of the three RGB components. If you specify only one gamma value, **pnmgamma** uses that value for all three RGB components.

If you don't specify any gamma parameters, **pnmgamma** chooses a default. For the transfer functions defi ned by standards, the default is the value defi ned by the standard. If you specify anything else, you will be varying from the standard. For the simple power function transfer function, the default gamma is 1/.45.

OPTIONS

-ungamma

Apply the inverse of the specifi ed transfer function (i.e. go from gamma-corrected nonlinear intensities to linear intensities).

-cieramp

Use the CIE Rec. 709 gamma transfer function. Note that it is true CIE Rec. 709 only if you use the default gamma value (i.e. don't specify any gamma parameters). This transfer function is a power function modifi ed with a linear ramp near black.

If you specify neither **-cieramp** nor **-srgbramp**, the transfer function defaults to a simple power function.

-srgbramp
Use the Internation Electrotechnical Commission (IEC) SRGB gamma transfer function (as specifi ed in the standard IEC 61966-2-1). Note that it is true SRGB only if you use the default gamma value (i.e. don't specify any gamma parameters). This transfer function is like the one selected by **-cieramp**, but with different constants in it.

Note that SRGB is often spelled "sRGB". In this document, we use standard English typography, though, which doesn't allow for that kind of capitalization.

If you specify neither **-cieramp** nor **-srgbramp**, the transfer function defaults to a simple power function.

WHAT IS GAMMA?

A good explanation of gamma is in Charles Poynton's GammaFAQ at <http://www.inforamp.net/˜ poynton/ColorFAQ.html> and ColorFAQ at <http://www.inforamp.net/˜ poynton/GammaFAQ.html>

In brief: The simplest way to code an image is by using sample values that are directly proportional to the intensity of the color components. But that wastes the sample space because the human eye can't discern differences between low-intensity colors as well as it can between high-intensity colors. So instead, we pass the light intensity values through a transfer function that makes it so that changing a sample value by 1 causes the same level of perceived color change anywhere in the sample range. We store those resulting values in the image fi le. That transfer function is called the gamma transfer function and the transformation is called gamma correcting.

Virtually all image formats, either specifi ed or de facto, use gamma-corrected values for their sample values.

What's really nice about gamma is that by coincidence, the inverse function that you have to do to convert the gamma-corrected values back to real light intensities is done automatically by CRTs. You just apply a voltage to the CRT's electron gun that is proportional to the gamma-corrected sample value, and the intensity of light that comes out of the screen is close to the intensity value you had before you applied the gamma transfer function!

And when you consider that computer video devices usually want you to store in video memory a value proportional to the signal voltage you want to go to the monitor, which the monitor turns into a proportional drive voltage on the electron gun, it is really convenient to work with gamma-corrected sample values.

SEE ALSO

pnm(5)

AUTHOR

Copyright (C) 1991 by Bill Davidson and Jef Poskanzer.

NAME

pnmhisteq − histogram equalise a portable anymap

SYNOPSIS

pnmhisteq [**−gray**] [**−rmap** *pgmfile*] [**−wmap** *pgmfile*] [**−verbose**] [*pnmfile*]

DESCRIPTION

pnmhisteq increases the contrast of a portable graymap or pixmap through the technique of *histogram equalisation*[1]. A histogram of the luminance of pixels in the map is computed, from which a transfer function is calculated which spreads out intensity levels around histogram peaks and compresses them at troughs. This has the effect of using the available levels of intensity more efficiently and thereby increases the detail visible in the image.

Mathematically, if *N[i]* is the number of pixels of luminosity *i* in the image and *T* is the total number of pixels, luminosity *j* is replaced by:

```
 j
---
\
 >  N[i] / T

---
i=0
```

If you're processing a related set of images, for example frames of an animation, it's generally best to apply the same intensity map to every frame, since otherwise you'll get distracting frame-to-frame changes in the brightness of objects. **pnmhisteq**'s **−wmap** option allows you to save, as a portable graymap, the luminosity map computed from an image (usually a composite of the images you intend to process created with **pnmcat**). Then, you can subsequently process each of the individual images using the luminosity map saved in the file, supplied with the **−rmap** option.

OPTIONS

−gray When processing a pixmap, only gray pixels (those with identical red, green, and blue values) are included in the histogram and modified in the output image. This is a special purpose option intended for images where the actual data are gray scale, with colour annotations you don't want modified. Weather satellite images that show continent outlines in colour are best processed using this option. The option has no effect when the input is a graymap.

−rmap *mapfile*
Process the image using the luminosity map specified by the portable graymap *mapfile.* The graymap, usually created by an earlier run of **pnmhisteq** with the **−wmap** option, contains a single row with number of columns equal to the *maxval* (greatest intensity) of the image. Each pixel in the image is transformed by looking up its luminosity in the corresponding column in the map file and changing it to the value given by that column.

−wmap *mapfile*
Creates a portable graymap, *mapfile,* containing the luminosity map computed from the histogram of the input image. This map file can be read on subsequent runs of **pnmhisteq** with the **−rmap** option, allowing a group of images to be processed with an identical map.

−verbose Prints the histogram and luminosity map on standard error.

All flags can be abbreviated to their shortest unique prefix.

BUGS

Histogram equalisation is effective for increasing the visible detail in scientific imagery and in some continuous-tone pictures. It is often too drastic, however, for scanned halftone images, where it does an excellent job of making halftone artifacts apparent. You might want to experiment with **pgnnorm**, **ppmnorm**, and **pnmgamma** for more subtle contrast enhancement.

The luminosity map file supplied by the **−rmap** option must have the same *maxval* as the input image. This is always the case when the map file was created by the **−wmap** option of **pnmhisteq**. If this restriction causes a problem, simply adjust the *maxval* of the map with **pnmdepth** to agree with the input image.

If the input is a PBM file (on which histogram equalisation is an identity operation), the only effect of passing the file through **pnmhisteq** will be the passage of time.

SEE ALSO

pgmnorm(1), **pnm**(5), **pnmcat**(1), **pnmdepth**(1), **pnmgamma**(1), **pnmnorm**(1)

[1] Russ, John C. The Image Processing Handbook. Boca Raton: CRC Press, 1992. Pages 105-110.

AUTHOR

NAME

pnmhistmap - draw a histogram for a PGM or PPM fi le

SYNOPSIS

pnmhistmap [**-black**] [**-white**] [**-max** *N*] [**-verbose**] [*pnmfi le*]

DESCRIPTION

Reads a portable anymap as input, although bitmap (PBM) input produces an error message and no image. Produces an image showing a histogram of the color (or gray) values in the input. A graymap (PGM) input produces a bitmap output. A pixmap (PPM) input produces pixmap output with three overlaid histograms: a red one for the red input, a green one for the green input, and a blue one for the blue input. The output is fi xed in size: 256 pixels wide by 200 pixels high.

OPTIONS

-black Ignores the count of black pixels when scaling the histogram.

-white Ignores the count of white pixels when scaling the histogram.

The -black and -white options, which can be used seperately or together, are useful for images with a large percentage of pixels whose value is zero or 255, which can cause the remaining histogram data to become unreadbaly small. Note that, for pixmap inputs, these options apply to all colors; if, for example, the input has a large number of bright-red areas, you will probably want to use the -white option.

-max N
Force the scaling of the histogram to use N as the largest-count value. This is useful for inputs with a large percentage of single-color pixels which are not black or white.

-verbose
Report the progress of making the histogram, including the largest-count value used to scale the output.

All fflags can be abbreviated to their shortest unique prefi x.

BUGS

Assumes maxval is always 255. Images with a smaller maxval will only use the lower-value side of the histogram. This can be overcome either by piping the input through "pnmdepth 255" or by cutting and scaling the lower-value side of the histogram. Neither is a particularly elegant solution.

Should allow the output size to be specifi ed.

SEE ALSO

pgmhist(1), ppmhist(1), pgm(5), ppm(5)

AUTHOR

Wilson H. Bent. Jr. (whb@usc.edu).

NAME

pnmindex - build a visual index of a bunch of anymaps

SYNOPSIS

pnmindex [**-size** *N*] [**-across** *N*] [**-colors** *N*] [**-black**] [**-title** *T*] [**-quant**|**-noquant**] *pnmfi le* ...

DESCRIPTION

pnmindex creates an index image containing thumbnail (small) versions of a bunch of PNM fi les you supply.

pnmindex labels each thumbnail and, optionally, contains a title.

OPTIONS

-size *N* The size of each thumbnail. The image is scaled to fi t maximally inside a *N* x *N* pixel box without changing its aspect ratio. Default is 100.

-across *N*
The number of thumbnails in each row. Default is 6.

-colors *N*
The maximum number of colors allowed in the overall image. If it would otherwise have more colors than these, **pnmindex** quantizes the result. The default is 256.

However, this value is meaningless if you specify the **-noquant** option.

-black This controls the color of the padding between the images; normally it's white and the labels are black lettering on white background, but the **-black** fla g reverses this.

-title *title*
Specifi es a title top place at the top of the image. Default is no title.

-quant Enables quanization (to the number of colors specifi ed by **-colors**). Quantization is on by default but you can disable it with **-noquant.**

-noquant
See **-quant**.

SEE ALSO

pnmscale(1), **pnmcat**(1), **pbmtext**(1), **ppmquant**(1), **pnm**(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer. **-title** and **-noquant** added 2000 by John Heidemann.

NAME

pnminvert - invert a portable anymap

SYNOPSIS

pnminvert [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Inverts it black for white and produces a portable anymap as output.

SEE ALSO

pnm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pnmmargin - add a border to a portable anymap

SYNOPSIS

pnmmargin [**-white**|**-black**|**-color** *colorspec*] *size* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Adds a border of the specified number of pixels, and produces a portable anymap as output.

OPTIONS

You can specify the border color with the **-white**, **-black**, and **-color** flags. If no color is specified, the program makes a guess.

SEE ALSO

pnm(5)

BUGS

It's a script. Scripts are not portable to non-Unix environments.

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

pnmmontage − create a montage of portable anymaps

SYNOPSIS

pnmmontage [**−?**|**−help**] [**−header**=*headerfile*] [**−quality**=*n*] [**−prefix**=*prefix*] [**−0**|**−1**|**−2**|**...**|**−9**] pnmfile...

DESCRIPTION

Packs images of differing sizes into a minimum-area composite image, optionally producing a C header file with the locations of the subimages within the composite image.

OPTIONS

−?, −help
Displays a (very) short usage message.

−header
Tells **pnmmontage** to write a C header file of the locations of the original images within the packed image. Each original image generates four #defines within the packed file: xxxX, xxxY, xxxSZX, and xxxSZY, where xxx is the name of the file, converted to all uppercase. The #defines OVERALLX and OVERALLY are also produced, specifying the total size of the montage image.

−prefix Tells **pnmmontage** to use the specified prefix on all of the #defines it generates.

−quality
Before attempting to place the subimages, **pnmmontage** will calculate a minimum possible area for the montage; this is either the total of the areas of all the subimages, or the width of the widest subimage times the height of the tallest subimage, whichever is greater. **pnmmontage** then initiates a problem-space search to find the best packing; if it finds a solution that is (at least) as good as the minimum area times the quality as a percent, it will break out of the search. Thus, **-q 100** will find the best possible solution; however, it may take a very long time to do so. The default is **-q 200.**

−0, −1, ... −9
These options control the quality at a higher level than **-q; −0** is the worst quality (literally pick the first solution found), while **−9** is the best quality (perform an exhaustive search of problem space for the absolute best packing). The higher the number, the slower the computation. The default is **−5.**

NOTES

Using **−9** is excessively slow on all but the smallest image sets. If the anymaps differ in maxvals, then pnmmontage will pick the smallest maxval which is evenly divisible by each of the maxvals of the original images.

SEE ALSO

pnmcat(1), **pnmindex**(1), **pnm**(5), **pam**(5), **pbm**(5), **pgm**(5), **ppm**(5)

AUTHOR

Copyright (C) 2000 by Ben Olmstead.

NAME

pnmnlfi lt - non-linear fi lters: smooth, alpha trim mean, optimal estimation smoothing, edge enhancement.

SYNOPSIS

pnmnlfi lt *alpha radius* [*pnmfi le*]

DESCRIPTION

pnmnlfi lt produces an output image where the pixels are a summary of multiple pixels near the corresponding location in an input image.

This program works on multi-image streams.

This is something of a swiss army knife fi lter. It has 3 distinct operating modes. In all of the modes each pixel in the image is examined and processed according to it and its surrounding pixels values. Rather than using the 9 pixels in a 3x3 block, 7 hexagonal area samples are taken, the size of the hexagons being controlled by the radius parameter. A radius value of 0.3333 means that the 7 hexagons exactly fi t into the center pixel (ie. there will be no fi ltering effect). A radius value of 1.0 means that the 7 hexagons exactly fi t a 3x3 pixel array.

Alpha trimmed mean fi lter. (0.0 <= alpha <= 0.5)

The value of the center pixel will be replaced by the mean of the 7 hexagon values, but the 7 values are sorted by size and the top and bottom alpha portion of the 7 are excluded from the mean. This implies that an alpha value of 0.0 gives the same sort of output as a normal convolution (ie. averaging or smoothing fi lter), where radius will determine the "strength" of the fi lter. A good value to start from for subtle fi ltering is alpha = 0.0, radius = 0.55 For a more blatant effect, try alpha 0.0 and radius 1.0

An alpha value of 0.5 will cause the median value of the 7 hexagons to be used to replace the center pixel value. This sort of fi lter is good for eliminating "pop" or single pixel noise from an image without spreading the noise out or smudging features on the image. Judicious use of the radius parameter will fi ne tune the fi ltering. Intermediate values of alpha give effects somewhere between smoothing and "pop" noise reduction. For subtle fi ltering try starting with values of alpha = 0.4, radius = 0.6 For a more blatant effect try alpha = 0.5, radius = 1.0

Optimal estimation smoothing. (1.0 <= alpha <= 2.0)

This type of fi lter applies a smoothing fi lter adaptively over the image. For each pixel the variance of the surrounding hexagon values is calculated, and the amount of smoothing is made inversely proportional to it. The idea is that if the variance is small then it is due to noise in the image, while if the variance is large, it is because of "wanted" image features. As usual the radius parameter controls the effective radius, but it probably advisable to leave the radius between 0.8 and 1.0 for the variance calculation to be meaningful. The alpha parameter sets the noise threshold, over which less smoothing will be done. This means that small values of alpha will give the most subtle fi ltering effect, while large values will tend to smooth all parts of the image. You could start with values like alpha = 1.2, radius = 1.0 and try increasing or decreasing the alpha parameter to get the desired effect. This type of fi lter is best for fi ltering out dithering noise in both bitmap and color images.

Edge enhancement. (-0.1 >= alpha >= -0.9)

This is the opposite type of fi lter to the smoothing fi lter. It enhances edges. The alpha parameter controls the amount of edge enhancement, from subtle (-0.1) to blatant (-0.9). The radius parameter controls the effective radius as usual, but useful values are between 0.5 and 0.9. Try starting with values of alpha = 0.3, radius = 0.8

Combination use.

The various modes of **pnmnlfi lt** can be used one after the other to get the desired result. For instance to turn a monochrome dithered image into a grayscale image you could try one or two passes of the smoothing fi lter, followed by a pass of the optimal estimation fi lter, then some subtle edge enhancement. Note that using edge enhancement is only likely to be useful after one of the non-linear fi lters (alpha trimmed mean or optimal estimation fi lter), as edge enhancement is the direct opposite of smoothing.

For reducing color quantization noise in images (ie. turning .gif fi les back into 24 bit fi les) you could try a pass of the optimal estimation fi lter (alpha 1.2, radius 1.0), a pass of the median fi lter (alpha 0.5, radius 0.55), and possibly a pass of the edge enhancement filter. Several passes of the optimal estimation filter with declining alpha values are more effective than a single pass with a large alpha value. As usual, there is a tradeoff between filtering effectiveness and loosing detail. Experimentation is encouraged.

References:

The alpha-trimmed mean filter is based on the description in IEEE CG&A May 1990 Page 23 by Mark E. Lee and Richard A. Redner, and has been enhanced to allow continuous alpha adjustment.

The optimal estimation filter is taken from an article "Converting Dithered Images Back to Gray Scale" by Allen Stenger, Dr Dobb's Journal, November 1992, and this article references "Digital Image Enhancement and Noise Filtering by Use of Local Statistics", Jong-Sen Lee, IEEE Transactions on Pattern Analysis and Machine Intelligence, March 1980.

The edge enhancement details are from pgmenhance(1), which is taken from Philip R. Thompson's "xim" program, which in turn took it from section 6 of "Digital Halftones by Dot Diffusion", D. E. Knuth, ACM Transaction on Graphics Vol. 6, No. 4, October 1987, which in turn got it from two 1976 papers by J. F. Jarvis et. al.

SEE ALSO

pgmenhance(1), pnmconvol(1), pnm(5)

BUGS

Integers and tables may overflow if PPM_MAXMAXVAL is greater than 255.

AUTHOR

Graeme W. Gill graeme@labtam.oz.au

NAME

pnmnorm - normalize the contrast in a Netbpm image

SYNOPSIS

pnmnorm [**-bpercent** *N* | **-bvalue** *N*] [**-wpercent** *N* | **-wvalue** *N*] [**-keephues**] [**-brightmax**]

[*ppmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or an equals sign between an option name and its value.

DESCRIPTION

Reads a PNM image (PBM, PGM, or PPM). Normalizes the contrast by forcing the lightest pixels to white, the darkest pixels to black, and linearly rescaling the ones in between; and produces the same kind of fi le as output. This is pretty useless for a PBM image.

The program fi rst determines a mapping of old brightness to new brightness. For each possible brightness of a pixel, the program determines a corresponding brightness for the output image.

Then for each pixel in the image, the program computes a color which has the desired output brightness and puts that in the output. With a color image, it is not always possible to compute such a color and retain any semblance of the original hue, so the brightest and dimmest pixels may only approximate the desired brightness.

Note that for a PPM image, this is different from separately normalizing the individual color components.

OPTIONS

By default, the darkest 2 percent of all pixels are mapped to black, and the lightest 1 percent are mapped to white. You can override these percentages by using the **-bpercent** and **-wpercent** fhgs, or you can specify the exact pixel values to be mapped by using the **-bvalue** and **-wvalue** fhgs. Appropriate numbers for the fhgs can be gotten from the *ppmhist* tool. If you just want to enhance the contrast, then choose values at elbows in the histogram; e.g. if value 29 represents 3% of the image but value 30 represents 20%, choose 30 for *bvalue*. If you want to lighten the image, then set *bvalue* to 0 and just fi ddle with *wvalue*; similarly, to darken the image, set *wvalue* to maxval and play with *bvalue*.

The **-keephues** option says to keep each pixel the same hue as it is in the input; just adjust its intensity. By default, **pnmnorm** normalizes contrast in each component independently (except that the meaning of the **-wpercent** and **-bpercent** options are based on the overall intensities of the colors, not each component taken separately). So if you have a color which is intensely red but dimly green, **pnmnorm** would make the red more intense and the green less intense, so you end up with a different hue than you started with.

If you specify **-keephues**, **pnmnorm** would likely leave this pixel alone, since its overall intensity is medium.

-keephues can cause clipping, because a certain color may be below a target intensity while one of its components is saturated. Where that's the case, **pnmnorm** uses the maximum representable intensity for the saturated component and the pixel ends up with less overall intensity, and a different hue, than it is supposed to have.

This option is meaningless on grayscale images.

Before March 2002, there was no **-keephues** option.

The **-brightmax** option says to use the intensity of the most intense RGB component of a pixel as the pixel's brightness. By default, **pnmnorm** uses the luminosity of the color as its brightness.

This option is meaningless on grayscale images.

Before March 2002, there was no **-brightmax** option.

SEE ALSO

ppmhist(1), **pgmhist**(1), **pnmgamma**(1), **ppmbrighten**(1), **ppmdim**(1), **pnm**(5)

NAME

pnmpad - add borders to portable anymap

SYNOPSIS

pnmpad [**-verbose**] [**-white**|**-black**]
[[[**-width=***width* [**-halign=***ratio*]] |
  [**-left=***leftpad*] [**-right=***rightpad*]]]
[[[**-height=***height* [**-valign=***ratio*]] |
  [**-top=***toppad*] [**-bottom=***botpad*]]]
[*pnmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or an equals sign between an option name and its value.

DESCRIPTION

Reads a PNM image as input. Outputs a PNM image that is the input image plus black or white borders of the sizes specifi ed.

If you just need to convert an image to a certain size regardless of the original dimensions, **pnmcut** with the **-pad** option may be a better choice.

OPTIONS

-verbose
Verbose output.

-white
-black (default)
Set pad color.

-left *leftpad*
-right *rightpad*
-top *toppad*
-bottom *bottompad*
Specify amount of padding in pixels.

-width *width*
Set desired width of image. Overrides **-left** and **-right** if specifi ed. If *width* is less than the actual image width, it is ignored. Use **pnmcut**(1) to cut off parts of images.

-height *height*
Set desired height of image. Overrides **-top** and **-bottom** if specifi ed. If *height* is less than the actual image height, it is ignored. Use **pnmcut**(1) to cut off parts of images.

-halign *ratio*
A real number between 0 and 1. Used in conjunction with **-width**, sets the alignment ratio between left padding and right padding.

Useful values:

0.0	- left aligned;
0.5	- center aligned (default);
1.0	- right aligned.

-valign *ratio*
A real number between 0 and 1. Used in conjunction with **-height**, sets the alignment ratio between bottom padding and top padding.

Useful values:

0.0	- bottom aligned;
0.5	- center aligned (default);
1.0	- top aligned.

HISTORY

Before February 2002,

pnmpad had a different option syntax which was less expressive and not like conventional Netpbm programs. That syntax is still understood by **pnmpad** for backward compatibility, but not documented or supported for future use.

SEE ALSO

pbmmake(1), **pnmpaste**(1), **pnmcut**(1), **pnmcrop**(1), **pbm**(5)

AUTHOR

NAME

pnmpaste - paste a rectangle into a portable anymap

SYNOPSIS

pnmpaste [**-replace**|**-or**|**-and** |**-xor**] *frompnmfile x y* [*intopnmfile*]

DESCRIPTION

Reads two portable anymaps as input. Inserts the first anymap into the second at the specified location, and produces a portable anymap the same size as the second as output. If the second anymap is not specified, it is read from stdin. The *x* and *y* can be negative, in which case they are interpreted relative to the right and bottom of the anymap, respectively.

This tool is most useful in combination with *pnmcut*. For instance, if you want to edit a small segment of a large image, and your image editor cannot edit the large image, you can cut out the segment you are interested in, edit it, and then paste it back in.

Another useful companion tool is *pbmmask*.

pnmcomp is, a more general tool, except that it lacks the "or," "and," and "xor" functions. **pnmcomp** allows you to specify an alpha mask in order to have only part of the inserted image get inserted. So the inserted pixels need not be a rectangle. You can also have the inserted image be translucent, so the resulting image is a mixture of the inserted image and the base image.

The optional flag specifies the operation to use when doing the paste. The default is **-replace**. The other, logical operations are only allowed if both input images are bitmaps. These operations act as if white is TRUE and black is FALSE.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmcomp(1), **pnmcut**(1), **pnminvert**(1), **pnmarith**(1), **pbmmask**(1), **pnm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmpsnr - compute the difference between two images (the PSNR)

SYNOPSIS

pnmpsnr [*pnmfile1*] [*pnmfile2*]

DESCRIPTION

Reads two PBM, PGM, or PPM files, or PAM equivalents, as input. Prints the peak signal-to-noise ratio (PSNR) difference between the two images. This metric is typically used in image compression papers to rate the distortion between original and decoded image.

If the inputs are PBM or PGM, **pnmpsnr** prints the PSNR of the luminance only. Otherwise, it prints the separate PSNRs of the luminance, and chrominance (Cb and Cr) components of the colors.

The PSNR of a given component is the ratio of the mean square difference of the component for the two images to the maximum mean square difference that can exist betwee any two images. It is expressed as a decibel value.

The mean square difference of a component for two images is the mean square difference of the component value, comparing each pixel with the pixel in the same position of the other image. For the purposes of this computation, components are normalized to the scale [0..1].

The maximum mean square difference is identically 1.

So the higher the PSNR, the closer the images are. A luminance PSNR of 20 means the mean square difference of the luminances of the pixels is 100 times less than the maximum possible difference, i.e. 0.01.

SEE ALSO

pnm(5)

NAME

pnmquant - quantize the colors in a Netpbm image to a smaller set

SYNOPSIS

pnmquant [**-center**|**-meancolor**|**-meanpixel**] [**-fbyd**|**-fs**] [**-nofbyd**|**-nofs**] [**-spreadbrightness**|**-spreadluminosity**] *ncolors* [*pnmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or equals signs between an option name and its value.

DESCRIPTION

Reads a PNM image as input. Chooses *ncolors* colors to best represent the image, maps the existing colors to the new ones, and writes a PNM image as output.

This program is simply a combination of **pnmcolormap** and **pnmremap**, where the colors of the input are remapped using a color map which is generated from the colors in that same input. The options have the same meaning as in those programs. See their documentation to understand **pnmquant**.

It is much faster to call **pnmcolormap** and **pnmremap** directly than to run **pnmquant**. **pnmquant** is just a convenience.

ppmquant is an older program which does the same thing as **pnmquant**, but on only PPM images. It is, however, faster than either **pnmquant** or **ppmcolormap**/**pnmremap**.

SEE ALSO

pnmcolormap(1), **pnmremap**(1), **ppmquantall**(1), **pnmdepth**(1), **ppmdither**(1), **ppmquant**(1), **pnm**(5)

NAME

pnmremap - replace colors in a PPM image with colors from another set

SYNOPSIS

pnmremap [**-fbyd**|**-fs**|**-nfbyd**|**-nofs**] [**-fi rstisdefault**] [**-verbose**] [**-mapfi le**=*mapfi le*] [**-missing-color**=*color*] [*pnmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or an equals sign between an option name and its value.

DESCRIPTION

pnmremap replaces the colors in an input image with those from a colormap you specify. Where a color in the input is not in the colormap, you have three choices: 1) choose the closest color from the colormap; 2) choose the fi rst color from the colormap; 3) use a color specifi ed by a command option. (In this latter case, if the color you specify is not in your color map, the output will not necessarily contain only colors from the colormap).

Two reasons to do this are: 1) you want to reduce the number of colors in the input image; and 2) you need to feed the image to something that can handle only certain colors.

To reduce colors, you can generate the colormap with **ppmcolormap**. Example:

```
ppmcolormap testimg.ppm 256 >colormap.ppm
ppmremap -map=colormap.ppm testimg.ppm
>reduced_testimg.ppm
```

To limit colors to a certain set, a typical example is to create an image for posting on the World Wide Web, where different browsers know different colors. But all browsers are supposed to know the 216 "web safe" colors which are essentially all the colors you can represent in a PPM image with a maxval of 5. So you can do this:

```
ppmcolors 5 >websafe.ppm
ppmremap -map=webafe.ppm testimg.ppm >websafe_testimg.ppm
```

The output image has the same type and maxval as the map fi le.

PARAMETERS

There is one parameter, which is required: The fi le specifcation of the input PNM fi le.

OPTIONS

-fbyd **-fs** **-nofbyd** **-nofs** These options determine whether Floyd-Steinberg dithering is done. Without Floyd-Steinberg, the selection of output color of a pixel is based on the color of only the corresponding input pixel. With Floyd-Steinberg, multiple input pixels are considered so that the average color of an area tends to stay more the same than without Floyd-Steinberg. For example, if you map an image with a black, gray, gray, and white pixel adjacent, through a map that contains only black and white, it might result in an output of black, black, white, white. Pixel-by-pixel mapping would instead map both the gray pixels to the same color.

-fs is a synomym for **-fbyd**. **-nofs** is a synonym for **-nofbyd**.

The default is **-nofbyd**.

-firstisdefault

This affects what happens with a pixel in the input image whose color is not in the map file. If you specify neither **-firstisdefault** nor **-missingcolor**, **pnmremap** chooses for the output the color in the map which is closest to the color in the input. With **-firstisdefault**, **pnmremap** instead uses the first color in the colormap.

If you specify **-firstisdefault**, the maxval of your input must match the maxval of your colormap.

-missingcolor=*color*

This affects what happens with a pixel in the input image whose color is not in the map file. If you specify neither **-firstisdefault** nor **-missingcolor**, **pnmremap** chooses for the output the color in the map which is closest to the color in the input. With **-missingcolor**, **pnmremap** uses *color*. *color* need not be in the colormap.

If you specify **-missingcolor**, the maxval of your input must match the maxval of your colormap.

-verbose

Display helpful messages about the mapping process.

SEE ALSO

pnmcolormap(1), **ppmcolors**(1), **pnmquant**(1), **ppmquantall**(1), **pnmdepth**(1), **ppmdither**(1), **ppmquant**(1), **ppm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer. Copyright (C) 2001 by Bryan Henderson.

NAME

pnmrotate - rotate a portable anymap by some angle

SYNOPSIS

pnmrotate [**-noantialias**] *angle* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Rotates it by the specified angle and produces a portable anymap as output. If the input file is in color, the output will be too, otherwise it will be grayscale. The angle is in degrees (floating point), measured counter-clockwise. It can be negative, but it should be between -90 and 90. Also, for rotations greater than 45 degrees you may get better results if you first use *pnmflip* to do a 90 degree rotation and then *pnmrotate* less than 45 degrees back the other direction

The rotation algorithm is Alan Paeth's three-shear method. Each shear is implemented by looping over the source pixels and distributing fractions to each of the destination pixels. This has an "anti-aliasing" effect - it avoids jagged edges and similar artifacts. However, it also means that the original colors or gray levels in the image are modified. If you need to keep precisely the same set of colors, you can use the **-noantialias** flag. This does the shearing by moving pixels without changing their values. If you want anti-aliasing and don't care about the precise colors, but still need a limited *number* of colors, you can run the result through *ppmquant*.

All flags can be abbreviated to their shortest unique prefix.

REFERENCES

"A Fast Algorithm for General Raster Rotation" by Alan Paeth, Graphics Interface '86, pp. 77-81.

SEE ALSO

pnmshear(1), pnmflip(1), pnm(5), ppmquant(1)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmscale - scale a PNM image

SYNOPSIS

pnmscale *scale_factor* [*pnmfi le*]
pnmscale -reduce *reduction_factor* [*pnmfi le*]
pnmscale [{**-xsize=***cols* | **-width=***cols* | **-xscale=***factor*}] [{**-ysize=***rows* | **-height=***rows* | **-yscale=***factor*}] [*pnmfi le*]
pnmscale -xysize *cols rows* [*pnmfi le*]
pnmscale -pixels *n* [*pnmfi le*]

Miscellaneous options:
-verbose -nomix

Minimum unique abbreviation of option is acceptable. You may use double hypens instead of single hyphen to denote options. You may use white space in place of the equals sign to separate an option name from its value.

DESCRIPTION

Reads a PBM, PGM, or PPM image as input, scales it by the specifi ed factor or factors and produces a PGM or PPM image as output. If the input fi le is in color (PPM), the output will be too, otherwise it will be grayscale (PGM). This is true even if the input is a black and white bitmap (PBM), because the process of scaling can turn a combination of black and white pixels into a gray pixel.

If you want PBM output, use **pgmtopbm** to convert **pnmscale**'s output to PBM. Also consider **pbmreduce**.

You can both enlarge (scale factor > 1) and reduce (scale factor < 1).

When you specify an absolute size or scale factor for both dimensions, **pnmscale** scales each dimension independently without consideration of the aspect ratio.

If you specify one dimension as a pixel size and don't specify the other dimension, **pnmscale** scales the unspecifi ed dimension to preserve the aspect ratio.

If you specify one dimension as a scale factor and don't specify the other dimension, **pnmscale** leaves the unspecifi ed dimension unchanged from the input.

If you specify the *scale_factor* parameter instead of dimension options, that is the scale factor for both dimensions. It is equivalent to **-xscale=***scale_factor* **-yscale=***scale_factor* .

Specifying the **-reduce** *reduction_factor* option is equivalent to specifying the *scale_factor* parameter, where *scale_factor* is the reciprocal of *reduction_factor*.

-xysize specifi es a bounding box. **pnmscale** scales the input image to the largest size that fi ts within the box, while preserving its aspect ratio.

-pixels specifi es a maximum total number of output pixels. **pnmscale** scales the image down to that number of pixels. If the input image is already no more than that many pixels, **pnmscale** just copies it as output; **pnmscale** does not scale up with **-pixels**.

If you enlarge by a factor of 3 or more, you should probably add a *pnmsmooth* step; otherwise, you can see the original pixels in the resulting image.

When the scale factor is not an integer (including all cases of scaling down), there are two ways to do the scaling. Which one **pnmscale** does is controlled by its **-nomix** option.

By default, **pnmscale** mixes the colors of adjacent pixels to produce output pixels that contain information from multiple input pixels. This makes the image look more like it would if it had infinite resolution. Note that it means the output may contain colors that aren't in the input at all.

But if you specify **-nomix**, **pnmscale** never mixes pixels. Each output pixel is derived from one input pixel. If you're scaling up, pixels get duplicated. If you're scaling down, pixels get omitted. Note that this means the image is rather distorted. If you scale up by 1.5 horizontally, for example, the even numbered input pixels are doubled in the output and the odd numbered ones are copied singly.

When the scale factor is an integer (which means you're scaling up), the **-nomix** option has no effect -- output pixels are always just N copies of the input pixels. In this case, though, consider using **pamstretch** instead of **pnmscale** to get the added pixels interpolated instead of just copied and thereby get a smoother enlargement.

pnmscale with **-nomix** is faster than without, but **pnmenlarge** is faster still. **pnmenlarge** works only on integer enlargements.

A useful application of **pnmscale** is to blur an image. Scale it down (without **-nomix)** to discard some information, then scale it back up using **pamstretch**.

Or scale it back up with **pnmscale** and create a "pixelized" image, which is sort of a computer-age version of blurring.

PRECISION

pnmscale uses floating point arithmetic internally. There is a speed cost associated with this. For some images, you can get the acceptable results (in fact, sometimes identical results) faster with **pnmscalefixed**, which uses fixed point arithmetic. **pnmscalefixed** may, however, distort your image a little. See **pnmscalefixed**'s man page for a complete discussion of the difference.

SEE ALSO

pnmscalefixed(1), **pamstretch**(1), **pbmreduce**(1), **pnmenlarge**(1), **pnmsmooth**(1), **pnmcut**(1), **pnm(5)**

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmscale - scale a PNM fi le quickly

DESCRIPTION

pnmscalefi xed is the same thing as **pnmscale** except that it uses fi xed point arithmetic internally instead of fbating point, which makes it run faster. In turn, it is less accurate and may distort the image.

Use the **pnmscale** man page with **pnmscalefi xed**. This man page only describes the difference.

pnmscalefi xed uses fi xed point 12 bit arithmetic. By contrast, **pnmscale** uses fbating point arithmetic which on most machines is probably 24 bit precision. This makes **pnmscalefi xed** run faster (30% faster in one experiment), but the imprecision can cause distortions at the right and bottom edges.

The distortion takes the following form: One pixel from the edge of the input is rendered larger in the output than the scaling factor requires. Consequently, the rest of the image is smaller than the scaling factor requires, because the overall dimensions of the image are always as requested. This distortion will usually be very hard to see.

pnmscalefi xed with the **-verbose** option tells you how much distortion there is.

The amount of distortion depends on the size of the input image and how close the scaling factor is to an integral 1/4096th.

If the scaling factor is an exact multiple of 1/4096, there is no distortion. So, for example doubling or halving an image causes no distortion. But reducing it or enlarging it by a third would cause some distortion. To consider an extreme case, scaling a 100,000 row image down to 50,022 rows would create an output image with all of the input squeezed into the top 50,000 rows, and the last row of the input copied into the bottom 22 rows of output.

pnmscalefi xed could probably be modifi ed to use 16 bit or better arithmetic without losing anything. The modifi cation would consist of a single constant in the source code. Until there is a demonstrated need for that, though, the Netpbm maintainer wants to keep the safety cushion afforded by the original 12 bit precision.

pnmscalefi xed does not have **pnmscale ’s -nomix** option.

NAME

pnmshear - shear a portable anymap by some angle

SYNOPSIS

pnmshear [**-noantialias**] *angle* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Shears it by the specified angle and produces a portable anymap as output. If the input file is in color, the output will be too, otherwise it will be grayscale. The angle is in degrees (floating point), and measures this:

```
+-------+  +-------+
|       |  |\       \
|  OLD  |  | \  NEW  \
|       |  |an\       \
+-------+  |gle+-------+
```

If the angle is negative, it shears the other way:

```
+-------+  |-an+-------+
|       |  |gl/       /
|  OLD  |  |e/  NEW  /
|       |  |/       /
+-------+  +-------+
```

The angle should not get too close to 90 or -90, or the resulting anymap will be unreasonably wide.

The shearing is implemented by looping over the source pixels and distributing fractions to each of the destination pixels. This has an "anti-aliasing" effect - it avoids jagged edges and similar artifacts. However, it also means that the original colors or gray levels in the image are modified. If you need to keep precisely the same set of colors, you can use the **-noantialias** flag. This does the shearing by moving pixels without changing their values. If you want anti-aliasing and don't care about the precise colors, but still need a limited *number* of colors, you can run the result through *ppmquant*.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmrotate(1), pnmflip(1), pnm(5), ppmquant(1)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmsmooth - smooth out an image

SYNOPSIS

pnmsmooth [**-size** *width height*] [**-dump** *dumpfi le*] [*pnmfi le*]

DESCRIPTION

Smooths out an image by replacing each pixel with the average of its width X height neighbors. It is implemented as a C progam that generates a PGM convolution matrix and then invokes *pnmconvol*.

OPTIONS

-size width height

Specifi es the size of the convolution matrix. Default size is a 3x3 matrix. Width and height sizes must be odd. Maximum size of convolution matrix is limited by the maximum value for a pixel such that (width * height * 2) must not exceed the maximum pixel value.

-dump dumpfi le

Generates and saves the convolution fi le only. Use of this option does not invoke pnmconvol.

SEE ALSO

pnmconvol(1), pnm(5)

AUTHORS

Copyright (C) 1989, 1991 by Jef Poskanzer.
Converted from script to C program December 1994 by Mike Burns (burns@chem.psu.edu).

NAME

pnmsplit - split a multi-image PNM file into multiple single-image files

SYNOPSIS

pnmsplit [*pnmfile*[*output_file_pattern*]]

DESCRIPTION

Reads a Netpbm file as input. Copies each image in the input into a separate file, in the same format.

pnmfile is the file specification of the input file, or **-** to indicate Standard Input. The default is Standard Input.

output_file_pattern tells how to name the output files. It is the file specification of the output file, except that the first occurence of "%d" in it is replaced by the image sequence number in unpadded ASCII decimal, with the sequence starting at 0. If there is no "%d" in the pattern, **pnmsplit** fails.

The default output file pattern is "image%d".

Note that to do the reverse operation (combining multiple single-image PNM files into a multi-image one), there is no special Netpbm program. Just use **cat**.

SEE ALSO

pnm(5), **cat**(1)

NAME

pnmtile - replicate a portable anymap into a specified size

SYNOPSIS

pnmtile *width height* [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Replicates it until it is the specified size, and produces a portable anymap as output.

SEE ALSO

pnm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

Name

pnmtoddif − Convert a portable anymap to DDIF format

Syntax

pnmtoddif pnmtoddif [-resolution x y] [pnmfi le [ddiffi le]]

OPTIONS

resolution x y	The horizontal and vertical resolution of the output image in dots per inch. Defaults to 78 dpi.
pnmfi le	The fi lename for the image fi le in pnm format. If this argument is omitted, input is read from stdin.
ddiffi le	The fi lename for the image fi le to be created in DDIF format. If this argument is omitted, the ddiffi le is written to standard output. It can only specifi ed if a pnmfi le is also specifi ed.

DESCRIPTION

`pnmtoddif` takes a portable anymap from standard input and converts it into a DDIF image fi le on standard output or the specifi ed DDIF fi le.

pbm format (bitmap) data is written as 1 bit DDIF, pgm format data (greyscale) as 8 bit greyscale DDIF, and ppm format data is written as 8,8,8 bit color DDIF. All DDIF image fi les are written as uncompressed. The data plane organization is interleaved by pixel.

In addition to the number of pixels in the width and height dimension, DDIF images also carry information about the size that the image should have, that is, the physical space that a pixel occupies. PBM-PLUS images do not carry this information, hence it has to be externally supplied. The default of 78 dpi has the benefi cial property of not causing a resize on most Digital Equipment Corporation color monitors.

AUTHOR

Burkhard Neidecker-Lutz
Digital Equipment Corporation, CEC Karlsruhe
neideck@nestvx.enet.dec.com

NAME

pnmtofiasco − Convert PNM file to FIASCO compressed file

SYNOPSIS

pnmtofiasco [*option*]... [*filename*]...

DESCRIPTION

pnmtofiasco compresses the named pbm, pgm, or ppm image files, or Standard Input if no file is named, and produces a FIASCO file on Standard Output.

OPTIONS

All option names may be abbreviated; for example, --optimize may be written --optim or --opt. For most options a one letter short option is provided. Mandatory or optional arguments to long options are mandatory or optional for short options, too. Both short and long options are case sensitive.

The basic options are:

−i *name*, **−−input-name=***name*

Compress the named images, not Standard Input. If *name* is **-**, read Standard Input. *name* has to be either an image filename or a template of the form:

prefix[start-end{+,-}step]suffix

Templates are useful when compressing video streams: e.g., if you specify the template **img0[12-01-2].pgm**, then **pnmtofiasco** compresses the images img012.pgm, img010.pgm, ..., img002.pgm.

If *name* is a relative path, **pnmtofiasco** searches for the image files in the current directory and in the (colon-separated) list of directories given by the environment variable **FIASCO_IMAGES**.

−o *output-file*, **−−output-name=***name*

Write FIASCO output to the named file, not to Standard Output.

If *name* is a relative path and the environment variable **FIASCO_DATA** is a (colon-separated) list of directories, then **pnmtofiasco** writes the output file to the first (writable) directory of this list. Otherwise, **pnmtofiasco** write it to the current directory.

−q *N*, **−−quality=***N*

Set quality of compression to *N*. Quality is 1 (worst) to 100 (best); default is 20.

−v, **−−version**

Print **pnmtofiasco** version number, then exit.

−V *N*, **−−verbose** *N*

Set level of verbosity to *N*. Level is 0 (no output at all), 1 (show progress meter), or 2 (show detailed compression statistics); default is 1.

−B *N*, **−−progress-meter** *N*

Set type of progress-meter to *N*. The following types are available; default is 1:

0: no progress meter

1: RPM style progress bar using 50 hash marks

2: percentage meter

−f *name*, **−−confi g=***name*
Load parameter fi le *name* to initialize the options of **pnmtofi asco**. See fi le **system.fi ascorc** for an example of the syntax. Options of **pnmtofi asco** are set by any of the following methods (in the specifi ed order):

1) Global ressource fi le **/etc/system.fi ascorc**

2) $HOME/.fi ascorc

3) command line

4) --confi g=*name*

−h, **−−info**
Print brief help, then exit.

−H, **−−help**
Print detailed help, then exit.

The options for advanced users are:

−b *name*, **−−basis-name=***name*
Preload compression basis *name* into FIASCO. The basis *name* provides the initial compression dictionary. Either use one of the fi les "small.fco", "medium.fco", or "large.fco" that come with **pnmtofi asco** or create a new ASCII basis fi le.

−z *N*, **−−optimize=***N* **Set optimization level to**
N. Level is 0 (fastes) to 3 (slowest); default is 1. Be warned, the encoding time dramatically increased when *N*=**2** or *N*=**3** while the compression performance only slightly improves.

−P, **−−prediction**
Use additional predictive coding. If this optimization is enabled then the image is compressed in two steps. In the fi rst step, a coarse approximation of the image is computed using large unichrome blocks. Finally, the delta image is computed and the prediction error is approximated using the standard FIASCO algorithm.

−D *N*, **−−dictionary-size=***N*
Set size of dictionary that is used when coding the luminance band to *N*; default is 10000, i.e., the dictionary is not restricted.

−C *N*, **−−chroma-dictionary=***N*
Set size of dictionary that is used when coding chroma bands to *N*; default is 40.

−Q *N*, **−−chroma-qfactor=***N*
Reduce the quality of chroma band compression *N*-times with respect to the user defi ned quality *q* of the luminance band compression (**−−quality**=*q*); default is 2.

−t *N*, **−−tiling-exponent=***N*
Subdivide the image into 2ˆ *N* tiles prior coding; default is 4, i.e. the image is subdivided into 16 tiles. The processing order of the individual tiles is defi ned by the option **−−tiling-method=***name***.**

−T *name***, −−tiling-method=***name*

Order the individual image tiles (the image is subdivided into; see option **−−tiling-exponent=***N*) by method *name*; default is "desc-variance".

desc-variance: Tiles with small variances are processed first.

asc-variance: Tiles with large variances are processed first.

desc-spiral: Tiles are process in spiral order starting in the middle.

asc-spiral: Tiles are process in spiral order starting at the border.

−−rpf-mantissa=*N*

Use *N* mantissa bits for quantized coefficients.

−−dc-rpf-mantissa=*N*

Use *N* mantissa bits for quantized DC coefficients.

−−rpf-range=*N*

Coefficients outside the quantization interval [-*N*,+*N*] are set to zero.

−−dc-rpf-range=*N*

DC coefficients outside the quantization interval [-*N*,+*N*] are set to zero.

Additional options for video compression are:

−s *N*, **−−smooth=***N*

Smooth decompressed reference frames along the partitioning borders by the given amount *N*. *N* is 0 (no smoothing) to 100; default is 70. This factor is stored in the FIASCO file.

−m *N*, **−−min-level=***N*

Start prediction (motion compensated prediction or additional prediction) on block level *N*; default is level 6. I.e., motion compensation is applied to all image blocks of at least 8x8 pixels (binary tree level *N*=6), 16x8 (*N*=7), 16x16 (*N*=8), etc.

−M *N*, **−−max-level=***N*

Stop prediction (motion compensated prediction or additional prediction) on block level *N*; default is level 10. I.e., motion compensation is applied to all image blocks of at most 16x16 pixels (*N*=8), 32x16 (*N*=9), 32x32 (*N*=10), etc.

−2, **−−half-pixel**

Use half pixel precise motion compensation.

−F *N*, **−−fps=***N*

Set number of frames per second to *N*. This value is stored in the FIASCO output file and is used in the decoder dfiasco(1) to control the framerate.

−p *type*, **−−pattern=***type*

Defines the type of inter frame compression which should be applied to individual frames of a video stream. *type* is a sequence of characters; default is "IPPPPPPPPP". Element **N** defines the type of predicting which should be used for frame **N**; the frame type pattern is periodically extended. Valid characters are:

I: intra frame, i.e., no motion compensated prediction is used at all.

P: predicted frame, i.e., a previously encoded frame is used for prediction (forward prediction).

B: bidirectional predicted frame, i.e., not only a previously shown frame but also a frame of the future is used for prediction (forward, backward or interpolated prediction).

−−cross-B-search
Instead of using exhaustive search the "Cross-B-Search" algorithm is used to find the best interpolated prediction of B-frames.

−−B-as-past-ref
Also use previously encoded B-frames when prediction the current frame. If this option is not set, only I- and P-frames are used to predict the current frame.

EXAMPLES

pnmtofiasco < foo.ppm >foo.wfa
Compress the still image "foo.ppm" to the FIASCO file "foo.wfa" using the default options.

pnmtofiasco -2 -p "IBBPBBPBB" -fps 15 -o video.wfa foo0*.ppm
Compress the video frames "foo0*.ppm" to the FIASCO file "video.wfa" using half pixel precise motion compensation at a frame rate of 15 frames per second. Intra frame 1 is used to predict P-frame 4, frames 1 and 4 are used to predict B-frames 2 and 3, and so on. Frame 10 is again an intra-frame.

FILES

/etc/system.fiascorc
The systemwide initialization file.
$HOME/**.fiascorc**
The personal initialization file.

ENVIRONMENT

FIASCO_IMAGES
Search path for image files. Default is "./".
FIASCO_DATA
Search and save path for FIASCO files. Default is "./".

SEE ALSO

fiascotopnm(1), **ppmtojpeg**(1), **pnmtojbig**(1), **ppmtogif**(1), **pnm**(5)

Ullrich Hafner, Juergen Albert, Stefan Frank, and Michael Unger. **Weighted Finite Automata for Video Compression**, IEEE Journal on Selected Areas In Communications, January 1998
Ullrich Hafner. **Low Bit-Rate Image and Video Coding with Weighted Finite Automata**, Ph.D. thesis, Mensch & Buch Verlag, ISBN 3-89820-002-7, October 1999.

AUTHOR

Ullrich Hafner <hafner@bigfoot.de>

NAME

pnmtofits - convert a portable anymap into FITS format

SYNOPSIS

pnmtofits [**−max** *f*] [**−min** *f*] [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Produces a FITS (Flexible Image Transport System) file as output. The resolution of the output file is either 8 bits/pixel, or 16 bits/pixel, depending on the value of maxval in the input file. If the input file is a portable bitmap or a portable graymap, the output file consists of a single plane image (NAXIS = 2). If instead the input file is a portable pixmap, the output file will consist of a three-plane image (NAXIS = 3, NAXIS3 = 3). A full description of the FITS format can be found in Astronomy & Astrophysics Supplement Series 44 (1981), page 363.

OPTIONS

Flags **−min** and **−max** can be used to set DATAMAX, DATAMIN, BSCALE and BZERO in the FITS header, but do not cause the data to be rescaled.

SEE ALSO

fitstopnm(1), pgm(5)

AUTHOR

Copyright (C) 1989 by Wilson H. Bent (whb@hoh-2.att.com), with modifications by Alberto Accomazzi (alberto@cfa.harvard.edu).

NAME

pnmtojbig − PNM to JBIG file converter

SYNOPSIS

pnmtojbg [*options*] [*input-file* | − [*output-file*]]

DESCRIPTION

Reads in a PBM or PGM format image from a file or standard input, compresses it, and outputs the image as a *JBIG* bi-level image entity (BIE) file.

JBIG is a highly effective lossless compression algorithm for bi-level images (one bit per pixel), which is particularly suitable for scanned document pages.

A *JBIG* encoded image can be stored in several resolutions (progressive mode). These resolution layers can be stored all in one single BIE or they can be stored in several separate BIE files. All resolution layers except the lowest one are stored merely as differences to the next lower resolution layer, because this requires less space than encoding the full image completely every time. Each resolution layer has twice the number of horizontal and vertical pixels than the next lower layer. *JBIG* files can also store several bits per pixel as separate bitmap planes, and **pnmtojbig** can read a PGM file and transform it into a multi-bitplane BIE.

OPTIONS

−
A single hyphen instead of an input file name will cause **pnmtojbg** to read the data from standard input instead from a file.

−q
Encode the image in one single resolution layer (sequential mode). This is usually the most efficient compression method. By default, the number of resolution layers is chosen automatically such that the lowest layer image is not larger than 640 × 480 pixels.

−x *number*
Specify the maximal horizontal size of the lowest resolution layer. The default is 640 pixels.

−y *number*
Specify the maximal vertical size of the lowest resolution layer. The default is 480 pixels.

−l *number*
Select the lowest resolution layer that will be written to the BIE. It is possible to store the various resolution layers of a *JBIG* image in progressive mode into different BIEs. Options **−l** and **−h** allow to select the resolution-layer interval that will appear in the created BIE. The lowest resolution layer has number 0 and this is also the default value. By default all layers will be written.

−h *number*
Select the highest resolution layer that will be written to the BIE. By default all layers will be written. See also option **−l.**

−b
Use binary values instead of Gray code words in order to encode pixel values in multiple bitplanes. This option has only an effect if the input is a PGM file and if more than one bitplane is produced. Note that the decoder has to make the same selection but cannot determine from the BIE, whether Gray or binary code words were used by the encoder.

−d *number*
Specify the total number of differential resolution layers into which the input image will be split in addition to the lowest layer. Each additional layer reduces the size of layer 0 by 50 %. This option overrides options **−x** and **−y** which are usually a more comfortable way of selecting the number of resolution layers.

−s *number*
The *JBIG* algorithm splits each image into a number of horizontal stripes. This option specifies that each stripe shall have *number* lines in layer 0. The default value is selected so that approximately 35 stripes will be used for the whole image.

−m *number*
Select the maximum horizontal offset of the adaptive template pixel. The *JBIG* encoder uses a number of neighbour pixels in order to get statistical a priori knowledge of the probability, whether the next pixel will be black or white. One single pixel out of this template of context neighbor pixels can be moved around. Especially for

dithered images it can be a significant advantage to have one neighbor pixel which has a distance large enough to cover the period of a dither function. By default, the adaptive template pixel can be moved up to 8 pixels away. This encoder supports up to 23 pixels, however as decoders are only required to support at least a distance of 16 pixels by the standard, no higher value than 16 for *number* is recommended in order to maintain interoperability with other *JBIG* implementations. The maximal vertical offset of the adaptive template pixel is always zero.

−t *number* Encode only the specified number of most significant bit planes. This option allows to reduce the depth of an input PGM file if not all bits per pixel are needed in the output.

−o *number* *JBIG* separates an image into several horizontal stripes, resolution layers and planes, were each plane contains one bit per pixel. One single stripe in one plane and layer is encoded as a data unit called stripe data entity (SDE) inside the BIE. There are 12 different possible orders in which the SDEs can be stored inside the BIE and *number* selects which one shall be used. The order of the SDEs is only relevant for applications that want to decode a *JBIG* file which has not yet completely arrived from e.g. a slow network connection. For instance some applications prefer that the outermost of the three loops (stripes, layers, planes) is over all layers so that all data of the lowest resolution layer is transmitted first.
The following values for *number* select these loop arrangements for writing the SDEs (outermost loop first):

0	planes, layers, stripes
2	layers, planes, stripes
3	layers, stripes, planes
4	stripes, planes, layers
5	planes, stripes, layers
6	stripes, layers, planes

All loops count starting with zero, however by adding 8 to the above order code, the layer loop can be reversed so that it counts down to zero and then higher resolution layers will be stored before lower layers. Default order is 3 which writes at first all planes of the first stripe and then completes layer 0 before continuing with the next layer and so on.

−p *number* This option allows to activate or deactivate various optional algorithms defined in the *JBIG* standard. Just add the numbers of the following options which you want to activate in order to get the *number* value:

4	deterministic prediction (DPON)
8	layer 0 typical prediction (TPBON)
16	diff. layer typ. pred. (TPDON)
64	layer 0 two-line template (LRLTWO)

Except for special applications (like communication with *JBIG* subset implementations) and for debugging purposes you will normally not want to change anything here. The default is 28, which provides the best compression result.

−c The adaptive template pixel movement is determined as suggested in annex C of the standard. By default the template change takes place directly in the next line which is most effective. However a few conformance test examples in the standard require the adaptive template change to be delayed until the first line of the next stripe. This option selects this special behavior, which is normally not required except in order to pass some conformance test suite.

−v After the BIE has been created, a few technical details of the created file will be listed (verbose mode).

FORMATS

Most of the format **pnmtojbig** creates is defined by the *JBIG* standard.

The standard, however, does not specify which values in the BIE mean white and which mean black. It contains a recommendation that for a single plane image zero mean background and one mean foreground, but the Netpbm formats have no concept of foreground and background. And the standard says nothing about values for multiple plane BIEs.

pnmtojbig follows Markus Kuhn's implementation of the standard in the **pbmtojbg** program that comes with his *JBIG* library: If the BIE is a single plane BIE, zero means white and one means black. If it is a multiple plane BIE, zero means black and the maximal value is white.

STANDARDS

This program implements the *JBIG* image coding algorithm as specified in ISO/IEC 11544:1993 and ITU-T T.82(1993).

AUTHOR

pnmtojbig is based on the *JBIG* library by Markus Kuhn, part of his **JBIG-KIT** package. The **pbmtojbg** program is part of the *JBIG-KIT* package. The most recent version of that library and tools set is freely available on the Internet from anonymous ftp server ftp.informatik.uni-erlangen.de in directory pub/doc/ISO/JBIG/.

pnmtojbig is part of the Netpbm package of graphics tools.

SEE ALSO

pbm(5),**pgm**(5),**jbigtopbm**(1)

LICENSE

If you use **pnmtojbig**, you are using various patents, particularly on its arithmetic encoding method, and in all probability you do not have a license from the patent owners to do so.

NAME

pnmtojpeg − convert PNM image to a JFIF ("JPEG") image

SYNOPSIS

pnmtojpeg [*options*] [*filename*]

DESCRIPTION

pnmtojpeg converts the named PBM, PGM, or PPM image file, or the standard input if no file is named, to a JFIF file on the standard output.

pnmtojpeg uses the Independent JPEG Group's JPEG library to create the output file. See **http://www.ijg.org** for information on the library.

"JFIF" is the correct name for the image format commonly known as "JPEG." Strictly speaking, JPEG is a method of compression. The image format using JPEG compression that is by far the most common is JFIF. There is also a subformat of TIFF that uses JPEG compression.

EXIF is an image format that is a subformat of JFIF (to wit, a JFIF file that contains an EXIF header as an APP1 marker). **pnmtojpeg** creates an EXIF image when you specify the **-exif** option.

OPTIONS

The basic options are:

−−exif=*filespec*

This option specifies that the output image is to be EXIF (a subformat of JFIF), i.e. it will have an EXIF header as a JFIF APP1 marker. The contents of that marker are the contents of the specified file. The special value − means to read the EXIF header contents from standard input. It is invalid to specify standard input for both the EXIF header and the input image.

The EXIF file starts with a two byte field which is the length of the file, including the length field, in pure binary, most significant byte first. The special value of zero for the length field means there is to be no EXIF header, i.e. the same as no **-exif** option. This is useful for when you convert a file from JFIF to PNM using **jpegtopnm**, then transform it, then convert it back to JFIF with **pnmtojpeg**, and you don't know whether or not it includes an EXIF header. **jpegtopnm** creates an EXIF file containing nothing but two bytes of zero when the input JFIF file has no EXIF header. Thus, you can transfer any EXIF header from the input JFIF to the output JFIF without worrying about whether an EXIF header actually exists.

The contents of the EXIF file after the length field are the exact byte for byte contents of the APP1 marker, not counting the length field, that constitutes the EXIF header.

−−quality=*n*

Scale quantization tables to adjust image quality. *n* is 0 (worst) to 100 (best); default is 75. (See below for more info.)

−−grayscale

−−greyscale

Create gray scale JFIF file. With this option, **pnmtojpeg** converts color input to gray scale. If you don't specify this option, The output file is in color format if the input is PPM, and grayscale format if the input is PBM or PGM.

In the PPM input case, even if all the colors in the image are gray, the output is in color format. Of course, the colors in it are still gray. The difference is that color format takes up a lot more space and takes longer to create and process.

−−optimize

Perform optimization of entropy encoding parameters. Without this, **pnmtojpeg** uses default encoding parameters. **−−optimize** usually makes the JFIF file a little smaller, but **pnmtojpeg** runs somewhat slower and needs much more memory. Image quality and speed of decompression are unaffected by **−−optimize**.

−−progressive
: Create a progressive JPEG file (see below).

−−comment=*text*
: Include a comment marker in the JFIF output, with comment text *text*. Without this option, there are no comment markers in the output.

The **−−quality** option lets you trade off compressed file size against quality of the reconstructed image: the higher the quality setting, the larger the JFIF file, and the closer the output image will be to the original input. Normally you want to use the lowest quality setting (smallest file) that decompresses into something visually indistinguishable from the original image. For this purpose the quality setting should be between 50 and 95; the default of 75 is often about right. If you see defects at **−−quality=75**, then go up 5 or 10 counts at a time until you are happy with the output image. (The optimal setting will vary from one image to another.)

−−quality=100 generates a quantization table of all 1's, minimizing loss in the quantization step (but there is still information loss in subsampling, as well as roundoff error). This setting is mainly of interest for experimental purposes. Quality values above about 95 are *not* recommended for normal use; the compressed file size goes up dramatically for hardly any gain in output image quality.

In the other direction, quality values below 50 will produce very small files of low image quality. Settings around 5 to 10 might be useful in preparing an index of a large image library, for example. Try **−−quality=2** (or so) for some amusing Cubist effects. (Note: quality values below about 25 generate 2-byte quantization tables, which are considered optional in the JFIF standard. **pnmtojpeg** emits a warning message when you give such a quality value, because some other JFIF programs may be unable to decode the resulting file. Use **−−baseline** if you need to ensure compatibility at low quality values.)

The **−−progressive** option creates a "progressive JPEG" file. In this type of JFIF file, the data is stored in multiple scans of increasing quality. If the file is being transmitted over a slow communications link, the decoder can use the first scan to display a low-quality image very quickly, and can then improve the display with each subsequent scan. The final image is exactly equivalent to a standard JFIF file of the same quality setting, and the total file size is about the same -- often a little smaller. **Caution:** progressive JPEG is not yet widely implemented, so many decoders will be unable to view a progressive JPEG file at all.

Options for advanced users:

−−dct=int
: Use integer DCT method (default).

−−dct=fast
: Use fast integer DCT (less accurate).

−−dct=float
: Use floating-point DCT method. The float method is very slightly more accurate than the int method, but is much slower unless your machine has very fast floating-point hardware. Also note that results of the floating-point method may vary slightly across machines, while the integer methods should give the same results everywhere. The fast integer method is much less accurate than the other two.

−−restart=*n*
: Emit a JPEG restart marker every *n* MCU rows, or every *n* MCU blocks if you append **B** to the number. **−−restart 0** (the default) means no restart markers.

−−smooth=*n*
: Smooth the input image to eliminate dithering noise. *n*, ranging from 1 to 100, indicates the strength of smoothing. 0 (the default) means no smoothing.

−−maxmemory=*n*
: Set a limit for amount of memory to use in processing large images. Value is in thousands of bytes, or millions of bytes if you append **M** to the number. For example, **−−max=4m** selects

4,000,000 bytes. If **pnmtojpeg** needs more space, it will use temporary files.

−−verbose
Print to the Standard Error file messages about the conversion process. This can be helpful in debugging problems.

The **−−restart** option tells **pnmtojpeg** to insert extra markers that allow a JPEG decoder to resynchronize after a transmission error. Without restart markers, any damage to a compressed file will usually ruin the image from the point of the error to the end of the image; with restart markers, the damage is usually confined to the portion of the image up to the next restart marker. Of course, the restart markers occupy extra space. We recommend **−−restart=1** for images that will be transmitted across unreliable networks such as Usenet.

The **−−smooth** option filters the input to eliminate fine-scale noise. This is often useful when converting dithered images to JFIF: a moderate smoothing factor of 10 to 50 gets rid of dithering patterns in the input file, resulting in a smaller JFIF file and a better-looking image. Too large a smoothing factor will visibly blur the image, however.

Options for wizards:

−−baseline
Force baseline-compatible quantization tables to be generated. This clamps quantization values to 8 bits even at low quality settings. (This switch is poorly named, since it does not ensure that the output is actually baseline JPEG. For example, you can use **−−baseline** and **−−progressive** together.)

−−qtables= *filespec*
Use the quantization tables given in the specified text file.

−−qslots=n[,...]
Select which quantization table to use for each color component.

−−sample=*HxV[,...]*
Set JPEG sampling factors for each color component.

−−scans= *filespec*
Use the scan script given in the specified text file. See below for information on scan scripts.

The "wizard" options are intended for experimentation with JPEG. If you don't know what you are doing, **don't use them**. These switches are documented further in the file wizard.doc that comes with the Independent JPEG Group's JPEG library.

EXAMPLES

This example compresses the PPM file foo.ppm with a quality factor of 60 and saves the output as foo.jpg:

```
pnmtojpeg −−quality=60 foo.ppm > foo.jpg

cat foo.bmp | bmptoppm | pnmtojpeg > foo.jpg
```

HINTS

JFIF is not ideal for cartoons, line drawings, and other images that have only a few distinct colors. For those, try instead **pnmtopng** or **ppmtobmp**. If you need to convert such an image to JFIF, though, you should experiment with **pnmtojpeg**'s **−−quality** and **−−smooth** options to get a satisfactory conversion. **−−smooth 10** or so is often helpful.

JPEG compression is notable for being a "lossy." This means that, unlike with most graphics conversions, you lose information, which means image quality, when you convert to JFIF. If you convert from PPM to JFIF and back repeatedly, image quality loss will accumulate. After ten or so cycles the image may be noticeably worse than it was after one cycle.

Because of this, you should do all the manipulation you have to do on the image in some other format and convert to JFIF as the last step. And if you can keep a copy in the original format, so much the better. PNG is a good choice for a format that is lossless, yet fairly compact. GIF is another way to go, but chances are you can't create a GIF image without owing a lot of money to Unisys and IBM, holders of patents on the LZW compression used in the GIF format.

The **−−optimize** option to **pnmtojpeg** is worth using when you are making a "final" version for posting or archiving. It's also a win when you are using low quality settings to make very small JFIF files; the percentage improvement is often a lot more than it is on larger files. (At present, **−−optimize** mode is automatically in effect when you generate a progressive JPEG file).

Another program, **cjpeg**, is similar. **cjpeg** is maintained by the Independent JPEG Group and packaged with the JPEG library which **pnmtojpeg** uses for all its JPEG work. Because of that, you may expect it to exploit more current JPEG features. Also, since you have to have the library to run **pnmtojpeg**, but not vice versa, **cjpeg** may be more commonly available.

On the other hand, **cjpeg** does not use the NetPBM libraries to process its input, as all the NetPBM tools such as **pnmtojpeg** do. This means it is less likely to be consistent with all the other programs that deal with the NetPBM formats. Also, the command syntax of **pnmtojpeg** is consistent with that of the other Netpbm tools, unlike **cjpeg**.

SCAN SCRIPTS

Use the **-scan** option to specify a scan script. Or use the **-progressive** option to specify a particular built-in scan script.

Just what a scan script is, and the basic format of the scan script file, is covered in the **wizard.doc** file that comes with the Independent JPEG Group's JPEG library. Scan scripts are same for **pnmtojpeg** as the are for **cjpeg**.

This section contains additional information that isn't, but probably should be, in that document.

First, there are many restrictions on what is a valid scan script. The JPEG library, and thus **pnmtojpeg**, checks thoroughly for any lack of compliance with these restrictions, but does little to tell you how the script fails to comply. The messages are very general and sometimes untrue.

To start with, the entries for the DC coefficient must come before any entries for the AC coefficients. The DC coefficient is Coefficient 0; all the other coefficients are AC coefficients. So in an entry for the DC coefficient, the two numbers after the colon must be 0 and 0. In an entry for AC coefficients, the first number after the colon must not be 0.

In a DC entry, the color components must be in increasing order. E.g. "0,2,1" before the colon is wrong. So is "0,0,0".

In an entry for an AC coeffient, you must specify only one color component. I.e. there can be only one number before the colon.

In the first entry for a particular coefficient for a particular color component, the "Ah" value must be zero, but the Al value can be any valid bit number. In subsequent entries, Ah must be the Al value from the previous entry (for that coefficient for that color component), and the Al value must be one less than the Ah value.

The script must ultimately specify at least some of the DC coefficent for every color component. Otherwise, you get the error message "Script does not transmit all the data." You need not specify all of the bits of the DC coefficient, or any of the AC coefficients.

There is a standard option in building the JPEG library to omit scan script capability. If for some reason your library was built with this option, you get the message "Requested feature was omitted at compile time."

ENVIRONMENT

JPEGMEM

If this environment variable is set, its value is the default memory limit. The value is specified as described for the **−−maxmemory** option. An explicit **−−maxmemory** option overrides any **JPEGMEM**.

SEE ALSO

cjpeg(1), **djpeg**(1), **jpegtran**(1), **rdjpgcom**(1), **wrjpgcom**(1)
ppm(5), **pgm**(5), **jpegtopnm**(1)
Wallace, Gregory K. "The JPEG Still Picture Compression Standard", Communications of the ACM, April 1991 (vol. 34, no. 4), pp. 30-44.

LIMITATIONS

Arithmetic coding is not supported for legal reasons.

The program could be much faster.

AUTHOR

pnmtojpeg and this man page were derived in large part from **cjpeg**, by the Independent JPEG Group. The program is otherwise by Bryan Henderson on March 07, 2000.

NAME

pnmtopalm - convert a portable anymap into a Palm pixmap

SYNOPSIS

pnmtopalm [**-verbose**] [**-depth** *N*] [**-maxdepth** *N*] [**-colormap**] [**-transparent** *color*] [**-offset**] [**-rle-compression**|**-scanline-compression**] [*pnmfile*]

DESCRIPTION

Reads a PNM image as input, from stdin or *pnmfile*. Produces a Palm pixmap as output.

Palm pixmap files are either greyscale files 1, 2, or 4 bits wide, or color files 8 bits wide, so **pnmtopalm** automatically scales colors to have an appropriate maxval, unless you specify a depth or max depth. Input files must have an appropriate number and set of colors for the selected output constraints. This often means that you should run the PNM image through **ppmquant** before you pass it to **pnmtopalm**. Netpbm comes with several colormap files you can use with **ppmquant** for this purpose. They are **palmgray2.map** (4 shades of gray for a depth of 2), **palmgray4.map** (16 shades of gray for a depth of 4), and **palmcolor8.map** (232 colors in default Palm colormap).

OPTIONS

-verbose
Display the format of the output file.

-depth *N*
Produce a file of depth *N*, where *N* must be either 1, 2, 4, 8, or 16. Any depth greater than 1 will produce a version 1 or 2 bitmap. Because the default Palm 8-bit colormap is not grayscale, if the input is a grayscale or monochrome pixmap, the output will never be more than 4 bits deep, regardless of the specified depth. Note that 8-bit color works only in PalmOS 3.5 (and higher), and 16-bit direct color works only in PalmOS 4.0 (and higher). However, the 16-bit direct color format is also compatible with the various PalmOS 3.x versions used in the Handspring Visor, so these images may also work in that device.

-maxdepth *N*
Produce a file of minimal depth, but in any case less than *N* bits wide. If you specify 16-bit, the output will always be 16-bit direct color.

-offset
Fill in the **nextDepthOffset** field in the file header, to provide for multiple renditions of the pixmap in the same file.

-colormap
Build a custom colormap and include it in the output file. This is not recommended by Palm, for efficiency reasons. Otherwise, **pnmtopalm** uses the default Palm colormap for color output.

-transparent *color*
Marks *one* particular color as fully transparent. The format to specify the color is either (when for example orange) "1.0,0.5,0.0", where the values are floats between zero and one, or with the syntax "#RGB", "#RRGGBB" or "#RRRRGGGGBBBB" where R, G and B are hexadecimal numbers. This also makes the output bitmap a version 2 bitmap. Transparency works only on Palm OS 3.5 and higher.

-rle-compression
Specifies that the output Palm bitmap will use the Palm RLE compression scheme, and will be a version 2 bitmap. RLE compression works only with Palm OS 3.5 and higher.

-scanline-compression
Specifies that the output Palm bitmap will use the Palm scanline compression scheme, and will be a version 2 bitmap. Scanline compression works only in Palm OS 2.0 and higher.

SEE ALSO

palmtopnm(1), **ppmquant**(1), **pnm**(5)

NOTES

An additional compression format, **packbits**, was added with PalmOS 4.0. This package should be updated to be able to generate that.

Palm pixmaps may contains multiple renditions of the same pixmap, in different depths. To construct an N-multiple-rendition Palm pixmap with **pnmtopalm**, first construct renditions 1 through N-1 using the **-offset** option, then construct the Nth pixmap without the **-offset** option. Then concatenate the individual renditions together in a single file using **cat**.

AUTHORS

This program was originally written as ppmtoTbmp.c, by Ian Goldberg and George Caswell. It was completely re-written by Bill Janssen to add color, compression, and transparency function.
Copyright 1995-2001 by Ian Goldberg, George Caswell, and Bill Janssen.

NAME

pnmtoplainpnm - convert portable any map to plain (ASCII) anymap format

SYNOPSIS

pnmtoplainpnm [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input, either from the named file or if no file named, from Standard Input. Writes out the image in plain (ASCII) anymap format to Standard Output. Of the three plain anymap formats, this program generates the one that corresponds naturally to the one of the three anymap formats that is the input (PBM for PBM, etc.).

SEE ALSO

pnm(5)

NAME

pnmtopng - convert a portable anymap into a Portable Network Graphics file

SYNOPSIS

pnmtopng [-verbose] [-downscale] [-interlace] [-alpha file]
[-transparent [=]color] [-background color] [-gamma value]
[-hist] [-chroma wx wy rx ry gx gy bx by] [-phys x y unit]
[-text file] [-ztxt file] [-time [yy]yy-mm-dd hh:mm:ss]
[-filter type] [-compression level] [-force] [*pnmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Portable Network Graphics file as output.

Color values in PNG files are either eight or sixteen bits wide, so *pnmtopng* will automatically scale colors to have a maxval of 255 or 65535. Grayscale files will be produced with bit depths 1, 2, 4, 8 or 16. An extra *pnmdepth* step is not necessary.

OPTIONS

-verbose
Display the format of the output file.

-downscale
Enables scaling of maxvalues of more then 65535 to 16 bit. Since this means loss of image data, the step is not performed by default.

-interlace
Creates an interlaced PNG file (Adam7).

-alpha file
The alpha channel of pixel (or image) specifies the transparency of a pixel. To create this fourth pixel value a separate *.pbm-* or *.pgm-file* is needed. In this file black (0) stands for fully transparant and white (1) will become opaque. The sizes of both pbm/pgm/ppm-files must be the same. If the information contained in the alpha mask can also be represented as a transparency index, it will be used, since this should result in a smaller image file.

-transparent color
ppmtogif marks the specified color as transparent in the PNG image.

You specify the color as in **ppmmake**(1).**E.g. red** or **rgb:ff/00/0d**. If the color you specify is not present in the image, **pnmtopnm** selects instead the color in the image that is closest to the one you specify. Closeness is measured as a cartesian distance between colors in RGB space. If multiple colors are equidistant, **pnmtopnm** chooses one of them arbitrarily.

However, if you prefix your color specification with "=", e.g.

-transparent =red

Only the exact color you specify will be transparent. If that color does not appear in the image, there will be no transparency. **pnmtopng** issues an information message when this is the case.

-background color
To create a background color chunck in the *png-file*, which can be used for subsequent alpha-channel or transparent-color conversions. See -transparent for format of color.

-gamma value
Creates an gAMA chunk. By providing the gamma-value of the *pnm-file* the software that lateron will display the *png-file* will be able to do the necessary gamma-corrections. A good rule-of-thumb is that when the file is created by a software program (like a CAD-program or a ray-tracer) the value is probably 1.0. When the *pnm-file* looks good on a non-gamma corrected PC display (which has itself a gamma-value of 2.2 - 2.8), a value of 0.45 should be given.

-hist Use this parameter to create a chunk that specifies the frequency (or histogram) of the colors in the image.

-chroma white point X and Y, red X and Y, green X and Y, and blue X and Y
To specify the white point and rgb values following the CIE-1931 spec.

-phys x y unit
When your image should not be displayed with square but with rectangular pixels this option should be used to create a pHYS chunk. When the unit-value is 0 the x and y only gives the ratio of pixel width and height. When it is 1 the x and y specify the number of pixels per meter.

-text file
Allows to include comments in the text-chunk of the *png-file.* The format of the text-file is as follows: when the first column does not contain a blank or a tab, the first word is considered to be the keyword. For keywords to contain spaces, enclose them in double-quotes.
When the first character on a line is a blank or tab, the rest of the line is a new line of the current comment. Note that the initial spaces are not considered to be part of the comment line.

Here is an example:

```
-------------------------------------------
Title           PNG-file
Author          your name
Description     how to include a text-chunk
                into a PNG file
"Creation date" 3-feb-1987
Software        pnmtopng
-------------------------------------------
```

-ztxt file
The same as -text, but now the text will be compressed.

-time yy-mm-dd hh:mm:ss or -time yyyy-mm-dd hh:mm:ss
This option allows you to specify the (modification)time. The year parameter can be given as a two- or a four-digit value.

-filter type
When the types of filters must be restricted you can specify here which filter you want to use. Allowed values are: 0 (none), 1 (sub), 2 (up), 3 (avg) and 4 (paeth).

-compression level
To explicitly set the compression level of zlib use this parameter. Select a level between 0 for no compression (max speed) and 9 for maximum compression.

-force When set, -force limits the optimizations of pnmtopng. A png-file similar to the pnm-input is as much as possible enforced. For example no paletted files will be created and alpha-channel images will not be converted to images with a transparency chunck.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

pngtopnm(1), gif2png(1), pnmgamma(1), pnm(5)

NOTE

Instead of xxxtopnm|pnmtopng, a specific converter should be used, if available. E.g. *gif2png* (GIF conversion), etc.

BUGS

There could be an option to read the comment text from pnm comments instead of a separate file.

The program could be much faster, with a bit of code optimizing.

AUTHORS

Copyright (C) 1995-1997 by Alexander Lehmann
and Willem van Schaik.

NAME

pnmtops - convert portable anymap to PostScript

SYNOPSIS

pnmtops [**-scale** *s*] [**-dpi** *n*] [**-imagewidth** *n*] [**-imageheight** *n*] [**-width=***N*] [**-height=***N*] [**-equalpixels**] [**-turn**|**-noturn**] [**-rle**|**-runlength**] [**-nocenter**] [**-nosetpage**] [*pnmfile*]

All options can be abbreviated to their shortest unique prefix. You may use two hyphens instead of one. You may separate an option name and its value with white space instead of an equals sign.

DESCRIPTION

Reads a Netpbm image as input. Produces Encapsulated PostScript as output.

If the input file is in color (PPM), **pnmtops** generates a color PostScript file. Some PostScript interpreters can't handle color PostScript. If you have one of these you will need to run your image through **ppmtopgm** first.

If you specify no output dimensioning options, the output image is dimensioned as if you had specified **-scale=1.0**, which means aproximately 72 pixels of the input image generate one inch of output (if that fits the page).

Use **-imagewidth**, **-imageheight**, **-equalpixels**, **-width**, **-height**, and **-scale** to adjust that.

OPTIONS

-imagewidth

-imageheight Tells how wide and high you want the image on the page, in inches. The aspect ratio of the image is preserved, so if you specify both of these, the image on the page will be the largest image that will fit within the box of those dimensions.

If these dimensions are greater than the page size, you get Postscript output that runs off the page.

You cannot use **imagewidth** or **imageheight** with **-scale** or **-equalpixels**.

-equalpixels

This option causes the output image to have the same number of pixels as the input image. So if the output device is 600 dpi and your image is 3000 pixels wide, the output image would be 5 inches wide.

You cannot use **-equalpixels** with **-imagewidth**, **-imageheight**, or **-scale**.

-scale tells how big you want the image on the page. The value is the number of inches of output image that you want 72 pixels of the input to generate.

But **pnmtops** rounds the number to something that is an integral number of output device pixels. E.g. if the output device is 300 dpi and you specify **-scale=1.0**, then 75 (not 72) pixels of input becomes one inch of output (4 output pixels for each input pixel). Note that the **-dpi** option tell **pnmtops** how many pixels per inch the output device generates.

If the size so specified does not fit on the page (as measured either by the **-width** and **-height** options or the default page size of 8.5 inches by 11 inches), **pnmtops** ignores the **-scale** option, issues a warning, and scales the image to fit on the page.

-dpi This option specifies the dots per inch of your output device. The default is 300 dpi. In theory PostScript is device-independent and you don't have to worry about this, but in practice its raster rendering can have unsightly bands if the device pixels and the image pixels aren't in sync.

Also this option is crucial to the working of the **equalpixels** option.

-width

-height These options specify the dimensions of the page on which the output is to be printed. This can affect the size of the output image.

The page size has no effect, however, when you specify the **-imagewidth**, **-imageheight**, or **-equalpixels** options.

These options may also affect positioning of the image on the page and even the paper selected (or cut) by the printer/plotter when the output is printed. See the **-nosetpage** option.

The default is 8.5 inches by 11 inches.

-turn **-noturn** These options control whether the image gets turned 90 degrees. Normally, if an image fi ts the page better when turned (e.g. the image is wider than it is tall, but the page is taller than it is wide), it gets turned automatically to better fi t the page. If you specify the **-turn** option, **pnmtops** turns the image no matter what its shape; If you specify **-noturn**, **pnmtops** does *not* turn it no matter what its shape.

-rle **-runlength** These identical options specify run-length compression. This may save time if the host-to-printer link is slow; but normally the printer's processing time dominates, so **-rle** makes things slower.

-nocenter

By default, **pnmtops** centers the image on the output page. You can cause **pnmtops** to instead put the image against the upper left corner of the page with the **-nocenter** option. This is useful for programs which can include PostScript fi les, but can't cope with pictures which are not positioned in the upper left corner.

For backward compatibility, **pnmtops** accepts the option **-center**, but it has no effect.

-nosetpage

pnmtops normally generates a "setpagedevice" directive to tell the printer/plotter what size paper to use (or cut). The dimensions it specifi es on this directive are those selected or defaulted by the **width** and **height** options or defaulted. If you don't want a "setpagedevice" directive in the output, specify **-nosetpage**. This can be useful if your printer chokes on this directive, which has not always been defi ned in Postscript, or you want to fake out the printer and print on one size paper as if you're printing on another.

SEE ALSO

pnm(5), **gs**(1), **psidtopgm**(1), **pstopnm**(1), **pbmtolps**(1), **pbmtoepsi**(1), **pbmtopsg3**(1), **ppmtopgm**(1),

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.
Modifi ed November 1993 by Wolfgang Stuerzlinger, wrzl@gup.uni-linz.ac.at

NAME

pnmtorast - convert a portable pixmap into a Sun rasterfile

SYNOPSIS

pnmtorast [**-standard**|**-rle**] [*pnmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Sun rasterfile as output.

Color values in Sun rasterfiles are eight bits wide, so *pnmtorast* will automatically scale colors to have a maxval of 255. An extra *pnmdepth* step is not necessary.

OPTIONS

The **-standard** flag forces the result to be in RT_STANDARD form; the **-rle** flag, RT_BYTE_ENCODED, which is smaller but, well, less standard. The default is **-rle**.

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

rasttopnm(1), pnm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnmtorle − convert a Netpbm image file into an RLE image file.

SYNOPSIS

pnmtorle [**−h**] [**−v**] [**−a**] [**−o** *outfile*] [*pnmfile*]

DESCRIPTION

This program converts Netpbm image files into Utah **RLE**(5) image files. You can include an alpha mask. If the input is a multiple image file, the output consists of several concatenated RLE images.

The RLE file will contain either a three channel color image (24 bits) or a single channel grayscale image (8 bits) depending upon the pnm file depth. If a converted ppm is displayed on an 8 bit display, the image must be dithered. In order to produce a better looking image (on 8 bit displays), it is recommended that the image be quantizing (to 8 bit mapped color) prior to its display. This may be done by piping the output of this program into the Utah **mcut**(1) or **rlequant**(1) utilities. An example of this is shown later.

OPTIONS

−v This option will cause pnmtorle to operate in verbose mode. The header information is written to "stderr". Actually, there is not much header information stored in a Netpbm file, so this information is minimal.

−h This option allows the header of the Netpbm image to be dumped to "stderr" without converting the file. It is equivalent to using the −v option except that no file conversion takes place.

−a This option causes pnmtorle to include an alpha channel in the output image. The alpha channel is based on the image: Wherever a pixel is black, the corresponding alpha value is transparent. Everywhere else, the alpha value is fully opaque.

−o *outfile*
If specified, the output will be written to this file. If *outfile* is "−", or if it is not specified, the output will be written to the standard output stream.

pnmfile The name of the Netpbm image data file to be converted. If not specified, standard input is assumed.

EXAMPLES

pnmtorle −v file.ppm −o file.rle
While running in verbose mode, convert file.ppm to RLE format and store resulting data in file.rle.

pnmtorle −h file.pgm
Dump the header information of the Netpbm file called file.pgm.

SEE ALSO

rletopnm(1), **urt**(1), **RLE**(5).

AUTHOR

Wes Barris
Army High Performance Computing Research Center (AHPCRC)
Minnesota Supercomputer Center, Inc.

NAME

pnmtosgi - convert a portable anymap to a SGI image file

SYNOPSIS

pnmtosgi [**-verbatim**|**-rle**] [**-imagename** *Name*] [**pnmfile**]

DESCRIPTION

Reads a portable anymap as input. Produces an SGI image file as output. The SGI image will be 2-dimensional (1 channel) for PBM and PGM input, and 3-dimensional (3 channels) for PPM.

OPTIONS

-verbatim
Write an uncompressed file.

-rle (default)
Write a compressed (run length encoded) file.

-imagename name
Write the string "name" into the imagename field of the header. The name string is limited to 79 characters. If no name is given, pnmtosgi writes "no name" into this field.

BUGS

Probably.

REFERENCES

SGI Image File Format documentation (draft v0.95) by Paul Haeberli (paul@sgi.com). Available via ftp at sgi.com:graphics/SGIIMAGESPEC.

SEE ALSO

pnm(5), sgitopnm(1)

AUTHOR

Copyright (C) 1994 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)

NAME

pnmtosir - convert a portable anymap into a Solitaire format

SYNOPSIS

pnmtosir [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Produces a Solitaire Image Recorder format.

pnmtosir produces an MGI TYPE 17 file for *pbm* and *pgm* files. For *ppm*, it writes a MGI TYPE 11 file.

SEE ALSO

sirtopnm(1), pnm(5)

BUGS

AUTHOR

Copyright (C) 1991 by Marvin Landis.

NAME

pnmtotiff - convert a PNM image to a TIFF file

SYNOPSIS

pnmtotiff [**-none**|**-packbits**|**-lzw**|**-g3**|**-g4**] [**-2d**] [**-fill**] [**-predictor** *n*] [**-msb2lsb**|**-lsb2msb**] [**-rowsperstrip** *n*] [**-minisblack**|**-miniswhite**] [**-truecolor**] [**-color**] [*pnmfile*]

Minimum unambiguous abbreviations of options are acceptable.

DESCRIPTION

Reads a PNM image as input. Produces a TIFF file as output.

The output goes to Standard Output, which must be a seekable file. That means no pipes, but any regular file should work.

OPTIONS

By default, **pnmtotiff** creates a TIFF file with no compression. This is your best bet most of the time. If you want to try another compression scheme or tweak some of the other even more obscure output options, there are a number of flags to play with.

Actually, the best default would be to use LZW compression, which is what **pnmtotiff** used to do by default. However, the Tiff library no longer does LZW compression due to concerns with violating Unisys's patent on LZW compression.

The **-none**, **-packbits**, **-lzw**, **-g3**, **-g4**, **-flate**, and **-adobeflat** options are used to override the default and set the compression scheme used in creating the output file. The CCITT Group 3 and Group 4 compression algorithms can only be used with bilevel data. **-lzw** doesn't really work because the Tiff library doesn't do LZW compression. It used to, but its developers removed the function out of concern about violating Unisys's patent. This option remains in case you use a Tiff library that cooperates, now or in the future. The **-2d** and **-fill** options are meaningful only with Group 3 compression: **-2d** requests 2-dimensional encoding, while **-fill** requests that each encoded scanline be zero-filled to a byte boundry. The **-predictor** option is only meaningful with LZW compression: a predictor value of 2 causes each scanline of the output image to undergo horizontal differencing before it is encoded; a value of 1 forces each scanline to be encoded without differencing.

By default, **pnmtotiff** creates a TIFF file with msb-to-lsb fill order. The **-msb2lsb** and **-lsb2msb** options are used to override the default and set the fill order used in creating the file.

The fill order is the order in which pixels are packed into a byte in the Tiff raster, in the case that there are multiple pixels per byte. msb-to-lsb means that the leftmost columns go into the most significant bits of the byte in the Tiff image. However, there is considerable confusion about the meaning of fill order. Some believe it means whether 16 bit sample values in the Tiff image are little-endian or big-endian. This is totally erroneous (The endianness of integers in a Tiff image is designated by the image's magic number). However, ImageMagick and older Netpbm both have been known to implement that interpretation. 2001.09.06.

If the image does not have sub-byte pixels, these options have no effect other than to set the value of the FILLORDER tag in the Tiff image (which may be useful for those programs that misinterpret the tag with reference to 16 bit samples).

The **-rowsperstrip** option can be used to set the number of rows (scanlines) in each strip of data in the output file. By default, the output file has the number of rows per strip set to a value that will ensure each strip is no more than 8 kilobytes long.

The **-minisblack** and **-miniswhite** option force the output image to have a "minimum is black" or "minimum is white" photometric, respectively. If you don't specify either, **pnmtotiff uses minimum is black except** when using Group 3 or Group 4 compression, in which case **pnmtotiff** follows CCITT fax standards and uses "minimum is white." This usually results in better compression and is generally preferred for bilevel coding.

Before February 2001, **pnmtotiff** always produced "minimum is black," due to a bug. In either case, **pnmtotiff** sets the photometric interpretation tag in the TIFF output according to which photometric is actually used.

-truecolor tells **pnmtotiff** to produce the 24-bit RGB form of TIFF output if it is producing a color TIFF image. Without this option, **pnmtotiff** produces a colormapped (paletted) 8-bit TIFF image unless there are more than 256 colors (and in the latter case, issues a warning).

The **-truecolor** option can prevent **pnmtotiff** from making two passes through the input file, thus improving speed and memory usage. See the section MULTIPLE PASSES.

If **pnmtotiff** produces a grayscale TIFF image, this option has no effect.

-color tells **pnmtotiff** to produce a color, as opposed to grayscale, TIFF image if the input is PPM, even if it contains only shades of gray. Without this option, **pnmtotiff** produces a grayscale TIFF image if the input is PPM and contains only shades of gray, and at most 256 shades. Otherwise, it produces a color TIFF output. For PBM and PGM input, **pnmtotiff** always produces grayscale TIFF output and this option has no effect.

The **-color** option can prevent **pnmtotiff** from making two passes through the input file, thus improving speed and memory usage. See the section MULTIPLE PASSES.

NOTES

There are myriad variations of the TIFF format, and this program generates only a few of them. **pnmtotiff** creates a grayscale TIFF file if its input is a PBM (monochrome) or PGM (grayscale) file. **pnmtotiff** also creates a grayscale file if it input is PPM (color), but there is only one color in the image. If the input is a PPM (color) file and there are 256 colors or fewer, but more than 1, **pnmtotiff** generates a color palette TIFF file. If there are more colors than that, **pnmtotiff** generates an RGB (not RGBA) single plane TIFF file. Use **pnmtotiffcmyk** to generate the cyan-magenta-yellow-black ink color separation TIFF format.

The number of bits per sample in the TIFF output is determined by the maxval of the PNM input. If the maxval is less than 256, the bits per sample in the output is the smallest number that can encode the maxval. If the maxval is greater than or equal to 256, there are 16 bits per sample in the output.

Multiple Passes

pnmtotiff reads the input image once if it can, and otherwise twice. It needs that second pass to analyze the colors in the image and generate a color map (pallette) and determine if the image is grayscale. So the second pass only happens when the input is PPM. And you can avoid it then by specifying both the **-truecolor** and **-color** options.

If the input image is small enough to fit in your system's file cache, the second pass is very fast. If not, it requires reading from disk twice, which can be slow.

When the input is from a file that cannot be rewound and reread, **pnmtotiff** reads the entire input image into a temporary file which can, and works from that. Even if it only needs one pass.

SEE ALSO

tifftopnm(1), **pnmtotiffcmyk**(1), **pnmdepth**(1), **pnm**(5)

AUTHOR

Derived by Jef Poskanzer from ras2tiff.c, which is Copyright (c) 1990 by Sun Microsystems, Inc. Author: Patrick J. Naughton (naughton@wind.sun.com).

NAME

pnmtotiffcmyk − convert a a portable anymap into a CMYK encoded TIFF file

SYNOPSIS

pnmtotiffcmyk [**Compargs**][**Tiffargs**][**Convargs**][*pnmfile*]

Compargs:

[**−none** | **−packbits** | **−lzw** [**−predictor** *n*]]

Tiffargs:

[**−msb2lsb** | **−lsb2msb**] [**−rowsperstrip** *n*]
[**−lowdotrange** *n*] [**−highdotrange** *n*]
[**−knormal** | **−konly** | **−kremove**]

Convargs:

[[**−default**][**Defargs**]| **−negative**]

Defargs:

[**−theta** *deg*] [**−gamma** *n*] [**−gammap** *-1*| **−gammap** *n*]

DESCRIPTION

Reads a portable anymap as input. Produces a CMYK encoded TIFF file as output. Optionally modifies the colour balance and black level, and removes CMY from under K.

OPTIONS

The order of most options is not important, but options for particular conversion algorithms must appear after the algorithm is selected (**−default**,**−negative**). If no algorithm is selected then **−default** is assumed and the appropriate options (**−theta**,**−gamma**,**−gammap**) can appear anywhere.

−none,**−packbits**,**−lzw**,**−predictor**

Tiff files can be compressed. By default LZW decompression is used, but (apparently) some readers cannot read this, so you may want to select a different algorithm (**−none**,**−packbits**). For LZW compression, a **−predictor** value of 2 forces horizontal differencing of scanlines before encoding; a value of 1 forces no differencing.

−msb2lsb,**−lsb2msb**

These flags control fill order (default is **-msb2lsb**).

−rowsperstrip

This sets the number of rows in an image strip (data in the Tiff files generated by this program is stored in strips - each strip is compressed individually). The default gives a strip size of no more than 8 kb.

−lowdotrange,**−highdotrange**

These options set tag values that may be useful for printers. They have not been tested.

−knormal,**−kremove**,**−konly**

These options modify the values written to the Tiff file after the conversion calculations (described below) are completed. They are useful only for testing and debugging the code.

−kremove sets the black (K) layer to zero while **−konly** sets all inks to the black value.

−default,**−negative**

−negative selects a simple algorithm that generates a colour negative. None of the following options apply to this algorithm, which is included as an example in the source to help implementors of other conversions. **−default** is not needed, unless it is used to countermand a **−negative** on the same command line. The default conversion from RGB to CMYK can be modified by altering the options listed below.

The CMYKTiff web site includes tests on the conversion parameters. The test images illustrate the command line options in practice and may make the following explanation clearer.

-theta *deg*

The basic conversion from RGB to CMY uses C = 1-R, M = 1-G, Y = 1-B. **−theta** provides a simple correction for any colour bias that may occur in the printed image because, in practice, inks do not exactly complement the primary colours. It rotates the colours by the amount given (*deg*) in degrees. Unless you are trying to produce unusual effects you will need to use small values (try generating three images at -10, 0 (the default) and 10 degrees and seeing which has the best colour balance.

−gamma *n*
The black (K) component of the image is calculated as min(C,Y,M). **−gamma** applies a gamma correction to this level. In other words, the final black level is K (normalised to the range 0 to 1) raised to the *n*th power. In practice this means that a value greater than 1 makes the image lighter and a value less than 1 makes the image darker. The range of allowed values is 0.1 to 10.

−gammap *n*
This option controls the removal of CMY under K. If *n* is -1 then no removal occurs and C, M, Y and K are calculated as above. This means that, when printed, dark areas contain all four inks, which can make high contrast areas, like lettering, appear fuzzy.

By default, when **−gammap** is not given on the command line, the colours are reduced in dark areas by subtracting the black level. The value subtracted is calculated with the same gamma correction given by **−gamma**. Hopefully this will reduce fuzziness without changing the appearance of the image significantly.

If **−gammap** *n* is given, with n between 0.01 and 10, then black is still subtracted, but the subtracted value is calculated using *n* rather than any value supplied with **−gamma**. For example, it may be best to only subtract black from the coloured inks in the very darkest regions. In that case, *n* should be a large value, such as 5.

BUGS

This program is not self-contained. It must be used with NetPbm and libtiff must be available (libtiff is included in the 1mar94 release of NetPbm).

SEE ALSO

pnmtotiff(1), tifftopnm(1), pnm(5)

AUTHOR

Copyright (c) 1999 Andrew Cooke (Jara Software). Released under the GPL with no warranty. See source or COPYRIGHT and LICENCE files in distribution for full details.

Much of the code (and man page!) uses ideas from other pnm programs, written by Jef Poskanzer (thanks go to him and libtiff maintainer Sam Leffler). A small section of the code - some of the tiff tag settings - is derived directly from pnmtotiff, by Jef Poskanzer, which, in turn, acknowledges Patrick Naughton with the following text:

Derived by Jef Poskanzer from ras2tif.c, which is:

Copyright (c) 1990 by Sun Microsystems, Inc.

Author: Patrick J. Naughton naughton@wind.sun.com

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

This file is provided AS IS with no warranties of any kind. The author shall have no liability with respect to the infringement of copyrights, trade secrets or any patents by this file or any part thereof. In no event will the author be liable for any lost revenue or profits or other special, indirect and consequential damages.

NAME

pnmtoxwd - convert a portable anymap into an X11 window dump

SYNOPSIS

pnmtoxwd [**-pseudodepth** *n*] [**-directcolor**] [*pnmfile*]

DESCRIPTION

Reads a portable anymap as input. Produces an X11 window dump as output. This window dump can be displayed using the xwud tool.

Normally, pnmtoxwd produces a StaticGray dump file for *pbm* and *pgm* files. For *ppm*, it writes a PseudoColor dump file if there are up to 256 colors in the input, and a DirectColor dump file otherwise. The **-directcolor** flag can be used to force a DirectColor dump. And the **-pseudodepth** flag can be used to change the depth of PseudoColor dumps from the default of 8 bits / 256 colors.

SEE ALSO

xwdtopnm(1), pnm(5), xwud(1)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

ppm3d - convert two portable pixmap into a red/blue 3d glasses pixmap

SYNOPSIS

ppm3d *leftppmfile rightppmfile* [*horizontal_offset*]

DESCRIPTION

Reads two portable pixmaps as input. Produces a portable pixmap as output, with the images overlapping by *horizontal_offset*

pixels in blue/red format.

horizontal_offset defaults to 30 pixels. Pixmaps MUST be the same size.

SEE ALSO

ppm(5)

AUTHOR

Copyright (C) 1993 by David K. Drum.

NAME

ppmbrighten - change an images Saturation and Value from an HSV map

SYNOPSIS

ppmbrighten [-n] [-s <+- saturation>] [-v <+- value>] <ppmfile>

DESCRIPTION

Reads a portable pixmap as input. Converts the image from RGB space to HSV space and changes the Value by <+- value> as a percentage. Likewise with the Saturation. Doubling the Value would involve

```
ppmbrighten -v 100
```

to add 100 percent to the Value.

The 'n' option normalizes the Value to exist between 0 and 1 (normalized).

SEE ALSO

pgmnorm(1), ppm(5)

AUTHOR

Copyright (C) 1990 by Brian Moffet Copyright (C) 1989 by Jef Poskanzer.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. This software is provided "as is" without express or implied warranty.

NOTES

This program does not change the number of colors.

NAME

ppmchange - change all pixels of one color to another in a portable pixmap

SYNOPSIS

ppmchange [**-closeness** *closeness_percent*] [**-remainder** *remainder_color*] [*oldcolor newcolor*] ... [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Changes all pixels of *oldcolor* to *newcolor*. You may specify up to 256 oldcolor/newcolor pairs on the command line. **ppmchange** leaves all colors not mentioned unchanged, unless you specify the **-remainder** option, in which case they are all changed to the single specified color.

You can specify that colors similar, but not identical, to the ones you specify get replaced by specifying a "closeness" factor.

The colors can be specified in five ways:

- A name, assuming that a pointer to an X11-style color names file was compiled in.
- An X11-style hexadecimal specifier: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.
- An X11-style decimal specifier: rgbi:r/g/b, where r g and b are floating point numbers between 0 and 1.
- For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.
- For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are floating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

  If a pixel matches two different *oldcolor*s, **ppmchange** replaces it with the *newcolor* of the leftmost specified one.

OPTIONS

-closeness *closeness_percent*

closeness is an integer per centage indicating how close to the color you specified a pixel must be to get replaced. By default, it is 0, which means the pixel must be the exact color you specified.

A pixel gets replaced if the distance in color between it and the color you specified is less than or equal to *closeness*.

The "distance" in color is defined as the cartesian sum of the individual differences in red, green, and blue intensities between the two pixels, normalized so that the difference between black and white is 100%.

This is probably simpler than what you want most the time. You probably would like to change colors that have similar chrominance, regardless of their intensity. So if there's a red barn that is variously shadowed, you want the entire barn changed. But because the shadowing significantly changes the color according to **ppmchange**'s distance formula, parts of the barn are probably about as distant in color from other parts of the barn as they are from green grass next to the barn.

Maybe **ppmchange** will be enhanced some day to do chrominance analysis.

-remainder *color*

ppmchange changes all pixels which are not of a color for which you specify an explicit replacement color on the command line to color *color*.

An example application of this is

ppmchange -remainder=black red red

to lift only the red portions from an image, or

ppmchange -remainder=black red white | ppmtopgm

to create a mask file for the red portions of the image.

SEE ALSO

pgmtoppm(1), **ppmcolormask**(1), **ppm**(5)

AUTHOR

Wilson H. Bent. Jr. (whb@usc.edu) with modifications by Alberto Accomazzi (alberto@cfa.harvard.edu)

NAME

ppmcie − draw a CIE color chart as a PPM image

SYNOPSIS

ppmcie [**−rec709**|**−cie**|**−ebu**|**−hdtv**|**−ntsc**|**−smpte**] [**−xy**|**−upvp**] [**−red** *rx ry*] [**−green** *gx gy*] [**−blue** *bx by*] [**−white** *wx wy*] [**−size** *edge*] [**−xsize**|**−width** *width*] [**−ysize**|**−height** *height*] [**−noblack**] [**−nowpoint**] [**−nolabel**] [**−noaxes**] [**−full**]

All options can be abbreviated to their shortest unique prefi x.

DESCRIPTION

ppmcie creates a PPM fi le containing a plot of the CIE ‘‘tongue’’ color chart -- to the extent possible in a PPM image. Alternatively, creates a pseudo-PPM image of the color tongue using RGB values from a color system of your choice.

The CIE color tongue is an image of all the hues that can be described by CIE X-Y chromaticity coordinates. They are arranged on a two dimensional coordinate plane with the X chromaticity on the horizontal axis and the Y chromaticity on the vertical scale. (You can choose alternatively to use CIE u’-v’ chromaticity coordinates, but the general idea of the color tongue is the same).

Note that the PPM format specifi es that the RGB values in the fi le are from CIE Rec. 709 color system, gamma-corrected. And positive. See **ppm**(5) for details. If you use one of the color system options on **ppmcie**, what you get is not a true PPM image, but is very similar. If you display such **ppmcie** output using a device that expects PPM input (which includes just about any computer graphics display program), it will display the wrong colors.

However, you may have a device that expects one of these variations on PPM.

In every RGB color system you can specify, including the default (which produces a true PPM image) there are hues in the color tongue that can’t be represented. For example, monochromatic blue-green with a wavelength of 500nm cannot be represented in a PPM image.

For these hues, **ppmcie** substitutes a similar hue as follows: They are desaturated and rendered as the shade where the edge of the Maxwell triangle intersects a line drawn from the requested shade to the white point defi ned by the color system’s white point. Furthermore, unless you specify the **-full** option, **ppmcie** reduces their intensity by 25% compared to the true hues in the image.

ppmcie draws and labels the CIE X-Y coordinate axes unless you choose otherwise with options.

ppmcie draws the Maxwell triangle for the color system in use on the color tongue. The Maxwell triangle is the triangle whose vertices are the primary illuminant hues for the color system. The hues inside the triangle show the color gamut for the color system. They are also the only ones that are correct for the CIE X-Y chromaticity coordinates shown. (See explanation above).

ppmcie also places a mark at the color system’s white point and displays in text the CIE X-Y chromaticities of the primary illuminants and white point for the color system. You can turn this off with options, though.

ppmcie annotates the periphery of the color tongue with the wavelength, in nanometers of the monochromatic hues which appear there.

Finally, **ppmcie** displays the black body chromaticity curve for Planckian radiators from 1000 to 30000 kelvins on the image.

You can choose from several standard color systems, or specify one of your own numerically.

CIE charts, by their very nature, contain a very large number of colors. If you’re encoding the chart for a color mapped device or file format, you'll need to use **ppmquant** or **ppmdither** to reduce the number of colors in the image.

OPTIONS

-rec709|-cie|-ebu|-hdtv|-ntsc|-smpte
: Select a standard color system whose gamut to plot. The default is **−rec709**, which chooses CIE Rec. 709, gamma-corrected. This is the only color system for which **ppmcie**'s output is a true PPM image. See explanation above. **−ebu** chooses the primaries used in the PAL and SECAM broadcasting standards. **−ntsc** chooses the primaries specified by the NTSC broadcasting system (few modern monitors actually cover this range). **−smpte** selects the primaries recommended by the Society of Motion Picture and Television Engineers (SMPTE) in standards RP-37 and RP-145, and **−hdtv** uses the much broader *HDTV ideal* primaries. **−cie** chooses a color system that has the largest possible gamut within the spectrum of the chart. This is the same color system as you get with the **-cie** option to John Walker's **cietoppm** program.

−xy plot CIE 1931 x y chromaticities. This is the default.

−upvp plot u' v' 1976 chromaticities rather than CIE 1931 x y chromaticities. The advantage of u' v' coordinates is that equal intervals of distance on the u' v' plane correspond roughly to the eye's ability to discriminate colors.

−red *rx ry* specifies the CIE *x* and *y* co-ordinates of the red illuminant of a custom color system and selects the custom system.

−green *gx gy*
: specifies the CIE *x* and *y* co-ordinates of the green illuminant of the color system and selects the custom system.

−blue *bx by*
: specifies the CIE *x* and *y* co-ordinates of the blue illuminant of the color system and selects the custom system.

−white *wx wy*
: specifies the CIE *x* and *y* co-ordinates of the white point of the color system and selects the custom system.

−size *edge* Create a pixmap of *edge* by *edge* pixels. The default is 512x512.

−xsize|−width *width*
: Sets the width of the generated image to *width* pixels. The default width is 512 pixels. If the height and width of the image are not the same, the CIE diagram will be stretched in the longer dimension.

−ysize|−height *height*
: Sets the height of the generated image to *height* pixels. The default height is 512 pixels. If the height and width of the image are not the same, the CIE diagram will be stretched in the longer dimension.

−noblack Don't plot the black body chromaticity curve.

−nowpoint
: Don't plot the color system's white point.

−nolabel Omit the label.

−noaxes Don't plot axes.

−full Plot the entire CIE tongue in full intensity; don't enhance the gamut of the specified color system.

SEE ALSO

ppmdither(1), **ppmquant**(1), **ppm**(5)

AUTHOR

NAME

ppmcolormask - produce mask of areas of a certain color in a PPM file

SYNOPSIS

ppmcolormask *color* [*ppmfile*]

DESCRIPTION

Reads a PPM file as input. Produces a PBM (bitmap) file as output. The output file is the same dimensions as the input file and is black in all places where the input file is the color *color*, and white everywhere else.

The output of **ppmcolormask** is useful as an alpha mask input to **pnmcomp**. Note that you can generate such an alpha mask automatically as you convert to PNG format with **pnmtopng**(1). Use its **-transparent** option.

ppmfile is the input file. If you don't specify *ppmfile*, the input is from Standard Input.

The output goes to Standard Output.

You can specify *color* five ways:

o An X11-style color name (e.g. **black**).

o An X11-style hexadecimal specifier: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.

o An X11-style decimal specifier: rgbi:r/g/b, where r g and b are floating point numbers between 0 and 1.

o For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.

o For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are floating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

SEE ALSO

pgmtoppm(1), **pnmcomp**(1), **pbmmask**(1), **ppm(5)**

AUTHOR

Bryan Henderson (bryanh@giraffe-data.com)

NAME

ppmcolors - generate a color map of all colors of a certain maxval

SYNOPSIS

ppmcolors [**-maxval=***maxval*]

All options can be abbreviated to their shortest unique prefix. You may use two hyphens instead of one to designate an option. You may use either white space or an equals sign between an option name and its value.

DESCRIPTION

ppmcolors generates a PPM color map containing all the colors representable with a certain maxval.

A PPM color map is a regular PPM image that is used by some programs to define a set of colors.

ppmcolors generates a one row PPM image that contains one pixel of each color representable by the maxval you choose. The maxval of the PPM image is that maxval. Note that you can change the maxval of the color map by running the output of **ppmcolors** through **pnmdepth**. As long as the new maxval is a multiple of the original, the resulting set of colors will be identical. If the new maxval is not a multiple, the resulting set of colors will be slightly different.

When you select a maxval of 5 (which is the default), you get a color map of the set of "web safe" colors as defined by Netscape. Most web browsers guarantee that they can produce at least these 216 colors (215 plus black).

pnmcolormap is another program to generate a color map. It chooses a set of colors designed to represent the colors in a specified image (or simply the set of all the colors in that image, if you choose).

pgmramp performs a similar function for PGM images.

OPTIONS

-maxval=*maxval*

This is the maxval of the generated color map. Default is 5.

SEE ALSO

pnmdepth(1), **pnmcolormap**(1), **ppmcie**(1), **ppmrainbow**(1), **pgmramp**(1), **ppm**(5)

AUTHOR

By Bryan Henderson, December 2001.

Contributed to the public domain.

NAME

ppmdim - dim a portable pixmap down to total blackness

SYNOPSIS

ppmdim *dimfactor* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Diminishes its brightness by the specified dimfactor down to total blackness. The dimfactor may be in the range from 0.0 (total blackness, deep night, nada, null, nothing) to 1.0 (original picture's brightness).

As *pnmgamma* does not do the brightness correction in the way I wanted it, this small program was written.

ppmdim is similar to *ppmbrighten* , but not exactly the same.

SEE ALSO

ppm(5), ppmflash(1), pnmgamma(1), ppmbrighten(1)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

ppmdist - simplistic grayscale assignment for machine generated, color images

SYNOPSIS

ppmdist [**-intensity**|**-frequency**] [*ppmfi le*]

DESCRIPTION

Reads a portable pixmap as input, performs a simplistic grayscale assignment intended for use with grayscale or bitmap printers.

Often conversion from ppm to pgm will yield an image with contrast too low for good printer output. The program maximizes contrast between the gray levels output.

A ppm input of n colors is read, and a pgm of n gray levels is written. The gray levels take on the values 0..n-1, while maxval takes on n-1.

The mapping from color to stepped grayscale can be performed in order of input pixel intensity, or input pixel frequency (number of repetitions).

OPTIONS

-frequency Sort input colors by the number of times a color appears in the input, before mapping to evenly distributed graylevels of output. **-intensity** Sort input colors by their grayscale intensity, before mapping to evenly distributed graylevels of output. This is the default.

BUGS

Helpful only for images with a very small number of colors. Perhaps should have been an option to ppmtopgm(1).

SEE ALSO

ppmtopgm(1), ppmhist(1), ppm(5)

AUTHOR

Copyright (C) 1993 by Dan Stromberg.

NAME

ppmdither - ordered dither for color images

SYNOPSIS

ppmdither [**-dim** *power*] [**-red** *shades*] [**-green** *shades*] [**-blue** *shades*] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input, and applies dithering to it to reduce the number of colors used down to the specified number of shades for each primary. The default number of shades is red=5, green=9, blue=5, for a total of 225 colors. To convert the image to a binary rgb format suitable for color printers, use -red 2 -green 2 -blue 2.

OPTIONS

-dim *power*	The size of the dithering matrix. The dithering matrix is a square whose dimension is a power of 2. *power* is that power of 2. The default is 4, for a 16 by 16 matrix.
-red *shades*	The number of red shades to be used, including black; minimum of 2.
-green *shades*	The number of green shades to be used, including black; minimum of 2.
-blue *shades*	The number of blue shades to be used, including black; minimum of 2.

SEE ALSO

pnmdepth(1), ppmquant(1), ppm(5)

AUTHOR

Copyright (C) 1991 by Christos Zoulas.

NAME

ppmfade − generate a transition between two image files using special effects.

SYNOPSIS

ppmfade [**-f** *first.ppm*] [**-l** *last.ppm*] [**-mix**|**-spread**|**-shift**|**-relief**|**-oil**|**-edge**|**-bentley**|**-block**] [**-base** *name*]

DESCRIPTION

This program generates a transition between either two input images or between one input image and black. You can use the 30 intermediate images generated to show a smooth transition between segments of a movie. The input and output images are in the Portable Pixmap (PPM) format. If you specify both input images, they should both be the same size. If you want to fade from black to an image, specify only the last image. If you want to fade from an image to black, specify only the first image. **ppmfade** names the resulting image files *base***.***nnnn* **.ppm**, where *nnnn* is a number varying between 0001 and 0030 and *base* is what you specify with via the **-base** option (default **fade**).

Another way to convert by steps from one image to another is morphing. You can use **xmorph** to do that.

OPTIONS

-f first.ppm
This is the image file (PPM format) to be used at the beginning of the transition. If not specified, the fade will start from black.

-l last.ppm
This is the image file (PPM format) to be used at the ending of the transition. If not specified, the fade will end with black.

-mix The two images are superimposed with the brightness of the first image decreasing from full to none and the brightness of the final image increasing from none to full. The transition is quadratic in brightness with faster transition in the beginning and slower at the end.

-spread
The pixels in the first image will be moved (spread) further and further from their original location and then moved into the proper location in the final image. This is the default transition.

-shift The pixels in the first image will be shifted further and further horizontally from their original location and then moved into the proper location in the final image.

-relief The first image is faded to a Laplacian relief filtered version of the first image. This is then faded to a Laplacian relief filtered version of the second image and finally faded to the final image.

-oil The first image is faded to an "oil transfer" version of the first image. This is then faded to an "oil transfer" version of the second image and finally faded to the final image.

-edge The first image is faded to an edge detected version of the first image. This is then faded to an edge detected version of the second image and finally faded to the final image.

-bentley
The first image is faded to a "Bentley Effect" version of the first image. This is then faded to a "Bentley Effect" version of the second image and finally faded to the final image.

-block The first image is defocused to small blocks. The small blocks are converted to match a defocused version of the last image. The block version of the last image is then focused to the final image.

-base*name*
This forms part of the output filenames, as described above.

EXAMPLES

```
ppmfade -f teapot.ppm -l pyr.ppm
```

Fade from teapot.ppm to pyr.ppm generating fade.0001.ppm to fade.0030.ppm using the "spread" transition.

ppmfade -l teapot.ppm

Fade from black to teapot.ppm generating fade.0001.ppm to fade.0030.ppm.

ppmfade -f teapot.ppm -base end

Fade from teapot.ppm to black generating end.0001.ppm to end.0030.ppm.

SEE ALSO

tontsc(1), **sgifade**(1), **smart_vfr**(1), **xmorph**(1), **ppm**(5),

AUTHOR

Wesley C. Barris (wesb@msc.edu)
Army High Performance Computing Research Center (AHPCRC)
Minnesota Supercomputer Center, Inc.

NAME

ppmflash - brighten a picture up to complete white-out

SYNOPSIS

ppmflash *flashfactor* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Increases its brightness by the specified flashfactor up to a total white-out image. The flashfactor may be in the range from 0.0 (original picture's brightness) to 1.0 (full white-out, The Second After).

As *pnmgamma* does not do the brightness correction in the way I wanted it, this small program was written.

This program is similar to *ppmbrighten* , but not exactly the same.

SEE ALSO

ppm(5), ppmdim(1), pnmgamma(1), ppmbrighten(1)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

ppmforge - fractal forgeries of clouds, planets, and starry skies

SYNOPSIS

ppmforge [**-clouds**] [**-night**] [**-dimension** *dimen*] [**-hour** *hour*] [**-inclination|-tilt** *angle*] [**-mesh** *size*] [**-power** *factor*] [**-glaciers** *level*] [**-ice** *level*] [**-saturation** *sat*] [**-seed** *seed*] [**-stars** *fraction*] [**-xsize|-width** *width*] [**-ysize|-height** *height*]

DESCRIPTION

ppmforge generates three kinds of ‘‘random fractal forgeries,’’ the term coined by Richard F. Voss of the IBM Thomas J. Watson Research Center for seemingly realistic pictures of natural objects generated by simple algorithms embodying randomness and fractal self-similarity. The techniques used by **ppmforge** are essentially those given by Voss[1], particularly the technique of spectral synthesis explained in more detail by Dietmar Saupe[2].

The program generates two varieties of pictures: planets and clouds, which are just different renderings of data generated in an identical manner, illustrating the unity of the fractal structure of these very different objects. A third type of picture, a starry sky, is synthesised directly from pseudorandom numbers.

The generation of planets or clouds begins with the preparation of an array of random data in the frequency domain. The size of this array, the ‘‘mesh size,’’ can be set with the **-mesh** option; the larger the mesh the more realistic the pictures but the calculation time and memory requirement increases as the square of the mesh size. The fractal dimension, which you can specify with the **-dimension** option, determines the roughness of the terrain on the planet or the scale of detail in the clouds. As the fractal dimension is increased, more high frequency components are added into the random mesh.

Once the mesh is generated, an inverse two dimensional Fourier transform is performed upon it. This converts the original random frequency domain data into spatial amplitudes. We scale the real components that result from the Fourier transform into numbers from 0 to 1 associated with each point on the mesh. You can further modify this number by applying a ‘‘power law scale’’ to it with the **-power** option. Unity scale leaves the numbers unmodifi ed; a power scale of 0.5 takes the square root of the numbers in the mesh, while a power scale of 3 replaces the numbers in the mesh with their cubes. Power law scaling is best envisioned by thinking of the data as representing the elevation of terrain; powers less than 1 yield landscapes with vertical scarps that look like glacially-carved valleys; powers greater than one make fairy-castle spires (which require large mesh sizes and high resolution for best results).

After these calculations, we have a array of the specifi ed size containing numbers that range from 0 to 1. The pixmaps are generated as follows:

Clouds A colour map is created that ranges from pure blue to white by increasing admixture (desaturation) of blue with white. Numbers less than 0.5 are coloured blue, numbers between 0.5 and 1.0 are coloured with corresponding levels of white, with 1.0 being pure white.

Planet The mesh is projected onto a sphere. Values less than 0.5 are treated as water and values between 0.5 and 1.0 as land. The water areas are coloured based upon the water depth, and land based on its elevation. The random depth data are used to create clouds over the oceans. An atmosphere approximately like the Earth’s is simulated; its light absorption is calculated to create a blue cast around the limb of the planet. A function that rises from 0 to 1 based on latitude is modulated by the local elevation to generate polar ice caps--high altitude terrain carries glaciers farther from the pole. Based on the position of the star with respect to the observer, the apparent colour of each pixel of the planet is calculated by ray-tracing from the star to the planet to the observer and applying a lighting model that sums ambient light and diffuse refl ection (for most planets ambient light is zero, as their primary star is the only source of illumination). Additional random data are used to generate stars around the planet.

Night A sequence of pseudorandom numbers is used to generate stars with a user specifi ed density.

Cloud pictures always contain 256 or fewer colours and may be displayed on most colour mapped devices without further processing. Planet pictures often contain tens of thousands of colours which must be compressed with **ppmquant** or **ppmdither** before encoding in a colour mapped format. If the display resolution is high enough, **ppmdither** generally produces better looking planets. **ppmquant** tends to create discrete colour bands, particularly in the oceans, which are unrealistic and distracting. The number of colours in starry sky pictures generated with the **-night** option depends on the value specified for **-saturation**. Small values limit the colour temperature distribution of the stars and reduce the number of colours in the image. If the **-saturation** is set to 0, none of the stars will be coloured and the resulting image will never contain more than 256 colours. Night sky pictures with many different star colours often look best when colour compressed by **pnmdepth** rather than **ppmquant** or **ppmdither**. Try *newmaxval* settings of 63, 31, or 15 with **pnmdepth** to reduce the number of colours in the picture to 256 or fewer.

OPTIONS

-clouds Generate clouds. A pixmap of fractal clouds is generated. Selecting clouds sets the default for fractal dimension to 2.15 and power scale factor to 0.75.

-dimension *dimen*
Sets the fractal dimension to the specified *dimen*, which may be any floating point value between 0 and 3. Higher fractal dimensions create more ‘‘chaotic’’ images, which require higher resolution output and a larger FFT mesh size to look good. If no dimension is specified, 2.4 is used when generating planets and 2.15 for clouds.

-glaciers *level*
The floating point *level* setting controls the extent to which terrain elevation causes ice to appear at lower latitudes. The default value of 0.75 makes the polar caps extend toward the equator across high terrain and forms glaciers in the highest mountains, as on Earth. Higher values make ice sheets that cover more and more of the land surface, simulating planets in the midst of an ice age. Lower values tend to be boring, resulting in unrealistic geometrically-precise ice cap boundaries.

-hour *hour*
When generating a planet, *hour* is used as the ‘‘hour angle at the central meridian.’’ If you specify **-hour 12**, for example, the planet will be fully illuminated, corresponding to high noon at the longitude at the centre of the screen. You can specify any floating point value between 0 and 24 for *hour*, but values which place most of the planet in darkness (0 to 4 and 20 to 24) result in crescents which, while pretty, don’t give you many illuminated pixels for the amount of computing that’s required. If no **-hour** option is specified, a random hour angle is chosen, biased so that only 25% of the images generated will be crescents.

-ice *level* Sets the extent of the polar ice caps to the given floating point *level*. The default level of 0.4 produces ice caps similar to those of the Earth. Smaller values reduce the amount of ice, while larger **-ice** settings create more prominent ice caps. Sufficiently large values, such as 100 or more, in conjunction with small settings for **-glaciers** (try 0.1) create ‘‘ice balls’’ like Europa.

-inclination|-tilt *angle*
The inclination angle of the planet with regard to its primary star is set to *angle*, which can be any floating point value from -90 to 90. The inclination angle can be thought of as specifying, in degrees, the ‘‘season’’ the planet is presently experiencing or, more precisely, the latitude at which the star transits the zenith at local noon. If 0, the planet is at equinox; the star is directly overhead at the equator. Positive values represent summer in the northern hemisphere, negative values summer in the southern hemisphere. The Earth’s inclination angle, for example, is about 23.5 at the June solstice, 0 at the equinoxes in March and September, and -23.5 at the December solstice. If no inclination angle is specified, a random value between -21.6 and 21.6 degrees is chosen.

-mesh *size* A mesh of *size* by *size* will be used for the fast Fourier transform (FFT). Note that memory requirements and computation speed increase as the square of *size*; if you double the mesh size, the program will use four times the memory and run four times as long. The default mesh is 256x256, which produces reasonably good looking pictures while using half a megabyte for the 256x256 array of single precision complex numbers required by the FFT. On machines with limited memory capacity, you may have to reduce the mesh size to avoid running out of RAM. Increasing the mesh size produces better looking pictures; the difference becomes particularly noticeable when generating high resolution images with relatively high fractal dimensions (between 2.2 and 3).

-night A starry sky is generated. The stars are created by the same algorithm used for the stars that surround planet pictures, but the output consists exclusively of stars.

-power *factor*
Sets the ‘‘power factor’’ used to scale elevations synthesised from the FFT to *factor*, which can be any floating point number greater than zero. If no factor is specified a default of 1.2 is used if a planet is being generated, or 0.75 if clouds are selected by the **-clouds** option. The result of the FFT image synthesis is an array of elevation values between 0 and 1. A non-unity power factor exponentiates each of these elevations to the specified power. For example, a power factor of 2 squares each value, while a power factor of 0.5 replaces each with its square root. (Note that exponentiating values between 0 and 1 yields values that remain within that range.) Power factors less than 1 emphasise large-scale elevation changes at the expense of small variations. Power factors greater than 1 increase the roughness of the terrain and, like high fractal dimensions, may require a larger FFT mesh size and/or higher screen resolution to look good.

-saturation *sat*
Controls the degree of colour saturation of the stars that surround planet pictures and fill starry skies created with the **-night** option. The default value of 125 creates stars which resemble the sky as seen by the human eye from Earth’s surface. Stars are dim; only the brightest activate the cones in the human retina, causing colour to be perceived. Higher values of *sat* approximate the appearance of stars from Earth orbit, where better dark adaptation, absence of skyglow, and the concentration of light from a given star onto a smaller area of the retina thanks to the lack of atmospheric turbulence enhances the perception of colour. Values greater than 250 create ‘‘science fiction’’ skies that, while pretty, don’t occur in this universe.

Thanks to the inverse square law combined with Nature’s love of mediocrity, there are many, many dim stars for every bright one. This population relationship is accurately reflected in the skies created by **ppmforge**. Dim, low mass stars live much longer than bright massive stars, consequently there are many reddish stars for every blue giant. This relationship is preserved by **ppmforge**. You can reverse the proportion, simulating the sky as seen in a starburst galaxy, by specifying a negative *sat* value.

-seed *num* Sets the seed for the random number generator to the integer *num*. The seed used to create each picture is displayed on standard output (unless suppressed with the **-quiet** option). Pictures generated with the same seed will be identical. If no **-seed** is specified, a random seed derived from the date and time will be chosen. Specifying an explicit seed allows you to re-render a picture you particularly like at a higher resolution or with different viewing parameters.

-stars *fraction*
Specifies the percentage of pixels, in tenths of a percent, which will appear as stars, either surrounding a planet or filling the entire frame if **-night** is specified. The default *fraction* is 100.

-xsize|-width *width*
Sets the width of the generated image to *width* pixels. The default width is 256 pixels. Images must be at least as wide as they are high; if a width less than the height is specified, it will be increased to equal the height. If you must have a long skinny pixmap, make a square one with **ppmforge**, then use **pnmcut** to extract a portion of the shape and size you require.

-ysize|-height *height*
Sets the height of the generated image to *height* pixels. The default height is 256 pixels. If the height specified exceeds the width, the width will be increased to equal the height.

All flags can be abbreviated to their shortest unique prefix.

BUGS

The algorithms require the output pixmap to be at least as wide as it is high, and the width to be an even number of pixels. These constraints are enforced by increasing the size of the requested pixmap if necessary.

You may have to reduce the FFT mesh size on machines with 16 bit integers and segmented pointer architectures.

SEE ALSO

pnmcut(1), **pnmdepth**(1), **ppmdither**(1), **ppmquant**(1), **ppm**(5)

[1] Voss, Richard F., ‘‘Random Fractal Forgeries,’’ in Earnshaw et. al., Fundamental Algorithms for Computer Graphics, Berlin: Springer-Verlag, 1985.

[2] Peitgen, H.-O., and Saupe, D. eds., The Science Of Fractal Images, New York: Springer Verlag, 1988.

AUTHOR

John Walker
Autodesk SA
Avenue des Champs-Montants 14b
CH-2074 MARIN
Suisse/Schweiz/Svizzera/Svizra/Switzerland
Usenet: kelvin@Autodesk.com
Fax: 038/33 88 15
Voice: 038/33 76 33

PLUGWARE! If you like this kind of stuff, you may also enjoy ‘‘James Gleick’s Chaos--The Software’’ for MS-DOS, available for $59.95 from your local software store or directly from Autodesk, Inc., Attn: Science Series, 2320 Marinship Way, Sausalito, CA 94965, USA. Telephone: (800) 688-2344 toll-free or, outside the U.S. (415) 332-2344 Ext 4886. Fax: (415) 289-4718. ‘‘Chaos--The Software’’ includes a more comprehensive fractal forgery generator which creates three-dimensional landscapes as well as clouds and planets, plus five more modules which explore other aspects of Chaos. The user guide of more than 200 pages includes an introduction by James Gleick and detailed explanations by Rudy Rucker of the mathematics and algorithms used by each program.

NAME

ppmhist - print a histogram of a PPM image

SYNOPSIS

ppmhist [**-hexcolor**] [**-noheader**] [**-map**] [**-nomap**] [**-sort=**{**frequency**,**rgb**}] [*ppmfi le*]

DESCRIPTION

Reads a PPM image as input. Generates a histogram of the colors in the image, i.e. a list of all the colors and how many pixels of each color are in the image.

OPTIONS

-sort={**frequency**,**rgb**}

The **-sort** option determines the order in which the colors are listed in the output. **frequency** means to list them in order of how pixels in the input image have the color, with the most represented colors fi rst. **rgb** means to sort them fi rst by the intensity of the red component of the color, the of the green, then of the blue, with the least intense fi rst.

The default is **frequency**.

-hexcolor

Print the color components in hexadecimal. Default is decimal.

-noheader

Do not print the column headings.

-map Generates a PPM fi le of the colormap for the image, with the color histogram as comments.

-nomap

Generates the histogram for human reading. This is the default.

SEE ALSO

ppm(5), **pgmhist**(1), **ppmtomap**(1), **pnmhistmap**(1), **ppmchange**(1)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

ppmlabel − add text to a portable pixmap

SYNOPSIS

ppmlabel [**−angle** *angle*] [**−background transparent** | *colour*] [**−colour** *colour*] [**−file** *filename*] [**−size** *textsize*] [**−text** *'text string'*] [**−x** *column*] [**−y** *row*] ... [*ppmfile*]

DESCRIPTION

ppmlabel uses the text drawing facilities of **ppmdraw** to add text to a portable pixmap. The location, size, baseline angle, colour of the text and background colour (if any) are controlled by command line arguments. The text can be specified on the command line or read from files. Any number of separate text strings can be added by one invocation of **ppmlabel**, limited only by the maximum length of the command line.

If no *ppmfile* is specified, **ppmdraw** reads its input pixmap from standard input.

OPTIONS

The arguments on the **ppmlabel** command line are not options in the strict sense; they are commands which control the placement and appearance of the text being added to the input pixmap. They are executed left to right, and any number of arguments may appear.

All flags can be abbreviated to their shortest unique prefix.

−angle *angle*
: Sets the angle of the baseline of subsequent text. *angle* is specified as an integral number of degrees, measured counterclockwise from the row axis of the pixmap.

−background transparent | *colour*
: If the argument is **"**transparent**",** text is drawn over the existing pixels in the pixmap. If a *colour* is given (see the **−colour** switch below for information on how to specify colours), rectangles enclosing subsequent text are filled with that colour.

−colour *colour*
: Sets the colour for subsequent text. The *colour* can be specified in five ways:

- A name, assuming that a pointer to an X11-style colour names file was compiled in.
- An X11-style hexadecimal specifier: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.
- An X11-style decimal specifier: rgbi:r/g/b, where r g and b are floating point numbers between 0 and 1.
- For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.
- For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are floating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

−file *filename*
: Reads text from the file *filename* and draws it on successive lines.

−size *textsize*
: Sets the height of the tallest characters above the baseline to *textsize* pixels.

−text *'text string'*
: Draws the given text string (which must be quoted if it contains spaces). The location for subsequent text is advanced by 1.75 times the current *textsize*, which allows drawing multiple lines of text in a reasonable manner without specifying the position of each line.

−x *column*
: Sets the column at which subsequent text will be left justified. Depending on the shape of the first character, the actual text may begin a few pixels to the right of this point.

−y *row*
: Sets the row which will form the baseline of subsequent text. Characters with descenders, such as "y", will extend below this line.

BUGS

Text strings are restricted to 7 bit ASCII. The text font used by **ppmdraw** doesn't include definitions for 8 bit ISO 8859/1 characters.

When drawing multiple lines of text with a non-transparent background, it should probably fill the space between the lines with the background colour. This is tricky to get right when the text is rotated to a non-orthogonal angle.

SEE ALSO

ppmmake(1), **ppm**(5)

AUTHOR

NAME

ppmmake - create a pixmap of a specifi ed size and color

SYNOPSIS

ppmmake *color width height*

DESCRIPTION

Produces a portable pixmap of the specifi ed color, width, and height.

The color can be specifi ed in fi ve ways:

- o A name, assuming that a pointer to an X11-style color names fi le was compiled in.
- o An X11-style hexadecimal specifi er: rgb:r/g/b, where r g and b are each 1- to 4-digit hexadecimal numbers.
- o An X11-style decimal specifi er: rgbi:r/g/b, where r g and b are fbating point numbers between 0 and 1.
- o For backwards compatibility, an old-X11-style hexadecimal number: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb.
- o For backwards compatibility, a triplet of numbers separated by commas: r,g,b, where r g and b are fbating point numbers between 0 and 1. (This style was added before MIT came up with the similar rgbi style.)

SEE ALSO

ppm(5), pbmmake(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

ppmmix - blend together two portable pixmaps

SYNOPSIS

ppmmix *fadefactor ppmfile1 ppmfile2*

DESCRIPTION

Reads two portable pixmaps as input. Mixes them together using the specified fade factor. The fade factor may be in the range from 0.0 (only ppmfile1's image data) to 1.0 (only ppmfile2's image data). Anything in between gains a smooth blend between the two images.

The two pixmaps must have the same size.

pnmcomp is a more general alternative. It allows you to mix images of different size and to have the fade factor vary throughout the image (through the use of an alpha mask).

SEE ALSO

pnmcomp(5), **ppm**(5)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

ppmntsc − Make RGB colors legal for NTSC or PAL color systems.

SYNOPSIS

ppmntsc [**−−pal**] [**−−legalonly**] [**−−illegalonly**] [**−−correctedonly**] [**−−verbose**] [**−−debug**] [*infile*]

Minimum unique abbreviations of options are acceptable.

DESCRIPTION

This program makes colors legal in the NTSC (or PAL) color systems. Often, images generated on the computer are made for use in movies which ultimately end up on video tape. However, the range of colors (as specified by their RGB values) on a computer does not match the range of colors that can be represented using the NTSC (or PAL) systems. If an image with "illegal" colors is sent directly to an NTSC (or PAL) video system for recording, the "illegal" colors will be clipped. This may result in an undesirable looking picture.

This utility tests each pixel in an image to see if it falls within the legal NTSC (or PAL) range. If not, it raises or lowers the pixel's saturation in the output so that it does fall within legal limits. Pixels that are already OK just go unmodified into the output.

Input is from the file named *input*. If *input* is **-**, input is from Standard Input. If you don't specify *input*, input is from Standard Input.

Output is always to Standard Output.

This program handles multi-image PPM input, producing multi-image PPM output.

OPTIONS

−−pal Use the PAL transform instead of the default NTSC.

−−verbose
Print a grand total of the number of illegal pixels.

−−debug
Produce a humongous listing of illegal colors and their legal counterparts. NOTE: This option may produce a great deal of output.

−−legalonly
Output only pixels that are already legal. Output black in place of pixels that are not.

−−illegalonly
Output only pixels that are illegal (and output them uncorrected). Output black in place of pixels that are already legal.

−−correctedonly
Output only pixels that are corrected versions of illegal pixels. Output black in place of pixels that are already legal.

SEE ALSO

ppm(5), **ppmdepth**(1), **ppmdim**(1), **ppmbrighten**(1)

AUTHOR

Wes Barris, Minnesota Supercomputer Center, Inc., Bryan Henderson

NAME

ppmpat - make a pretty pixmap

SYNOPSIS

ppmpat -gingham2|**-g2**|**-gingham3**| **-g3**|**-madras**|**-tartan**| **-poles**|**-squig**|**-camo**| **-anticamo** *width height*

DESCRIPTION

Produces a portable pixmap of the specifi ed width and height, with a pattern in it.

This program is mainly to demonstrate use of the ppmdraw routines, a simple but powerful drawing library. See the ppmdraw.h include fi le for more info on using these routines. Still, some of the patterns can be rather pretty. If you have a color workstation, something like **ppmpat -squig 300 300 | ppmquant 128** should generate a nice background.

OPTIONS

The different flags specify various different pattern types:

-gingham2
A gingham check pattern. Can be tiled.

-gingham3
A slightly more complicated gingham. Can be tiled.

-madras
A madras plaid. Can be tiled.

-tartan A tartan plaid. Can be tiled.

-poles Color gradients centered on randomly-placed poles. May need to be run through *ppmquant*.

-squig Squiggley tubular pattern. Can be tiled. May need to be run through *ppmquant*.

-camo Camouflage pattern. May need to be run through *ppmquant*.

-anticamo
Anti-camouflage pattern - like -camo, but ultra-bright colors. May need to be run through *ppmquant*.

All flags can be abbreviated to their shortest unique prefi x.

REFERENCES

Some of the patterns are from "Designer's Guide to Color 3" by Jeanne Allen.

SEE ALSO

pnmtile(1), ppmquant(1), ppm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

ppmquant - quantize the colors in a portable pixmap down to a specified number

SYNOPSIS

ppmquant [**-fbyd**|**-fs**] *ncolors* [*ppmfile*]
ppmquant [**-fbyd**|**-fs**] [**-nofbyd**|**-nofs**] **-mapfile** *mapfile* [*ppmfile*]

All options can be abbreviated to their shortest unique prefix. You may use two hyphens instead of one to designate an option. You may use either white space or equals signs between an option name and its value.

DESCRIPTION

pnmquant is a newer, more general program that is backward compatible with **ppmquant**. **ppmquant** may be faster, though.

Reads a PPM image as input. Chooses *ncolors* colors to best represent the image, maps the existing colors to the new ones, and writes a PPM image as output.

The quantization method is Heckbert's "median cut".

Alternately, you can skip the color-choosing step by specifying your own set of colors with the **-mapfile** option. The *mapfile* is just a *ppm* file; it can be any shape, all that matters is the colors in it. For instance, to quantize down to the 8-color IBM TTL color set, you might use:

```
P3
8 1
255
  0   0   0
255   0   0
  0 255   0
  0   0 255
255 255   0
255   0 255
  0 255 255
255 255 255
```

If you want to quantize one image to use the colors in another one, just use the second one as the mapfile. You don't have to reduce it down to only one pixel of each color, just use it as is.

If you use a mapfile, the output image has the same maxval as the mapfile. Otherwise, the output maxval is the same as the input maxval, or less in some cases where the quantization process reduces the necessary resolution.

The **-fbyd**/**-fs** option enables a Floyd-Steinberg error diffusion step. Floyd-Steinberg gives vastly better results on images where the unmodified quantization has banding or other artifacts, especially when going to a small number of colors such as the above IBM set. However, it does take substantially more CPU time, so the default is off.

-nofbyd/**-nofs** means not to use the Floyd-Steinberg error diffusion. This is the default.

REFERENCES

"Color Image Quantization for Frame Buffer Display" by Paul Heckbert, SIGGRAPH '82 Proceedings, page 297.

SEE ALSO

pnmquant(1), **ppmquantall**(1), **pnmdepth**(1), **ppmdither**(1), **ppm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

ppmquantall - run ppmquant on a bunch of files all at once, so they share a common colormap

SYNOPSIS

ppmquantall [**-ext** *extension*] *ncolors ppmfile* ...

DESCRIPTION

Takes a bunch of portable pixmap as input. Chooses *ncolors* colors to best represent all of the images, maps the existing colors to the new ones, and **overwrites the input files** with the new quantized versions.

If you don't want to overwrite your input files, use the **-ext** option. The output files are then named the same as the input files, plus a period and the extension text you specify.

Verbose explanation: Let's say you've got a dozen pixmaps that you want to display on the screen all at the same time. Your screen can only display 256 different colors, but the pixmaps have a total of a thousand or so different colors. For a single pixmap you solve this problem with *ppmquant*; this script solves it for multiple pixmaps. All it does is concatenate them together into one big pixmap, run *ppmquant* on that, and then split it up into little pixmaps again.

(Note that another way to solve this problem is to pre-select a set of colors and then use *ppmquant*'s **-map** option to separately quantize each pixmap to that set.)

SEE ALSO

ppmquant(1), **ppm(5)**

BUGS

It's a csh script. Csh scripts are not portable to System V. Scripts in general are not portable to non-Unix environments.

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

ppmqvga - 8 plane quantization

SYNOPSIS

ppmqvga [options] [input file]

DESCRIPTION

ppmqvga quantizes PPM files to 8 planes, with optional Floyd-Steinberg dithering. Input is a PPM file from the file named, or standard input of no file is provided.

Options

-d dither. Apply Floyd-Steinberg dithering to the data

-q quiet. Produces no progress reporting, and no terminal output unless and error occurs.

-v verbose. Produces additional output describing the number of colors found, and some information on the resulting mapping. May be repeated to generate loads of internal table output, but generally only useful once.

EXAMPLES

```
ppmqvga -d my_image.ppm | ppmtogif >my_image.gif

tgatoppm zombie.tga | ppmqvga | ppmtotif > zombie.tif
```

SEE ALSO

ppmquant

DIAGNOSTICS

Error messages if problems, various levels of optional progress reporting.

LIMITATIONS

none known.

AUTHOR

Original by Lyle Rains (lrains@netcom.com) as ppmq256 and ppmq256fs combined, documented, and enhanced by Bill Davidsen (davidsen@crd.ge.com)

Copyright

Copyright 1991,1992 by Bill Davidsen, all rights reserved. The program and documentation may be freely distributed by anyone in source or binary format. Please clearly note any changes.

NAME

ppmrainbow - Generate a rainbow

SYNOPSIS

ppmrainbow [**-width=***number*] [**-height=***number*]
[**-tmpdir=***directory*] [**-verbose**] *color ...*

All options can be abbreviated to their shortest unique prefix. You may use two hyphens instead of one to designate an option. You may use either white space or equals signs between an option name and its value.

DESCRIPTION

ppmrainbow generates a PPM image that fades from one color to another to another from left to right, like a rainbow. The colors are those you specify on the command line, in that order. The first color is added again on the right end of the image.

If you want a vertical or other non-horizontal rainbow, run the output through **pnmrotate**.

One use for such a rainbow is to compose it with another image under an alpha mask in order to add a rainbow area to another image. In fact, you can make rainbow-colored text by using **pbmtext**, **pnmcomp**, and **ppmrainbow**.

OPTIONS

-width *number*
The width in pixels of the output image.

Default is 600.

-height *number*
The height in pixels of the output image.

Default is 8.

-tmpdir
The directory specification of the directory **ppmrainbow** is to use for temporary files.

Default is the value of the **TMPDIR** environment variable, or */tmp* if **TMPDIR** is not set.

-verbose
Print the "commands" (invocations of other Netpbm programs) that **ppmrainbow** uses to create the image.

SEE ALSO

ppmmake(1), **pnmcomp**(1), **pbmtext**(1), **ppmfade**(1), **ppm**(5).

AUTHOR

Arjen Bax wrote **ppmrainbow** in June 2001 and contributed it to the Netpbm package. Bryan Henderson wrote this man page in July 2001.

NAME

ppmrelief - run a Laplacian relief filter on a portable pixmap

SYNOPSIS

ppmrelief [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Does a Laplacian relief filter, and writes a portable pixmap as output.

The Laplacian relief filter is described in "Beyond Photography" by Holzmann, equation 3.19. It's a sort of edge-detection.

SEE ALSO

pgmbentley(1), pgmoil(1), ppm(5)

AUTHOR

Copyright (C) 1990 by Wilson Bent (whb@hoh-2.att.com)

NAME

ppmshadow - add simulated shadows to a portable pixmap image

SYNOPSIS

ppmshadow [**−b** *blur_size*] [**−k**] [**−t**] [**−x** *xoffset*] [**−y** *yoffset*] [**−u**] [*pnmfi le*]

DESCRIPTION

ppmshadow adds a simulated shadow to an image, giving the appearance that the contents of the image fbat above the page, casting a diffuse shadow on the background. Shadows can either be black, as cast by opaque objects, or translucent, where the shadow takes on the colour of the object which casts it. You can specify the extent of the shadow and its displacement from the image with command line options.

OPTIONS

−b *blur_size*

Sets the distance of the light source from the image. Larger values move the light source closer, casting a more diffuse shadow, while smaller settings move the light further away, yielding a sharper shadow. *blur_size* defaults to 11 pixels.

−k Keep the intermediate temporary image fi les. When debugging, these intermediate fi les provide many clues as to the source of an error. See **FILES** below for a list of the contents of each fi le.

−t Consider the non-background material in the image translucent -- it casts shadows of its own colour rather than a black shadow, which is default. This often results in fuzzy, diffi cult-to-read images but in some circumstances may look better.

−u Print command syntax and a summary of options.

−x *xoffset*

Specifi es the displacement of the light source to the left of the image. Larger settings of **xoffset** displace the shadow to the right, as would be cast by a light further to the left. If not specifi ed, the horizontal offset is half of *blur_size* (above), to the left.

−y *yoffset*

Specifi es the displacement of the light source above the top of the image. Larger settings displace the shadow downward, corresponding to moving the light further above the top of the image. If you don't specify **−y**, the vertical offset defaults to the same as the horizontal offset (above), upward.

FILES

Input is an anymap named by the *pnmfi le* command line argument; if you don't specify *pnmfi le*, the input is the Standard Input fi le.

Output is a always a PPM fi le, written to Standard Output.

pnmfi le creates a number of temporary fi les as it executes. It creates them in the . directory, with names of the form:

_PPMshadow*pid*-*N***.ppm**

where *pid* is the process number of the **ppmshadow** process and *N* is a number identifying the fi le as described below. In normal operation, **ppmshadow** deletes temporary fi les as soon as it is done with them and leaves no debris around after it completes. To preserve the intermediate fi les for debugging, use the **−k** command line option.

N in the fi lename means:

1	Positive binary mask
2	Convolution kernel for blurring shadow
3	Blurred shadow image
4	Clipped shadow image, offset as requested
5	Blank image with background of source image
6	Offset shadow
7	Inverse mask file
8	Original image times inverse mask
9	Generated shadow times positive mask
10	Shadow times background colour

LIMITATIONS

The source image must contain sufficient space on the edges in the direction in which the shadow is cast to contain the shadow -- if it doesn't some of the internal steps may fail. You can usually expand the border of a too-tightly-cropped image with **pnmmargin** before processing it with **ppmshadow**.

Black pixels and pixels with the same color as the image background don't cast a shadow. If this causes unintentional "holes" in the shadow, fill the offending areas with a color which differs from black or the background by RGB values of 1, which will be imperceptible to the viewer. Since the comparison is exact, the modified areas will now cast shadows.

The background color of the source image (which is preserved in the output) is deemed to be the color of the pixel at the top left of the input image. If that pixel isn't part of the background, simply add a one-pixel border at the top of the image, generate the shadow image, then delete the border from it.

If something goes wrong along the way, the error messages from the various Netpbm programs **ppmshadow** calls will, in general, provide little or no clue as to where **ppmshadow** went astray. In this case, Specify the **−k** option and examine the intermediate results in the temporary files (which this option causes to be preserved). If you manually run the commands that **ppmshadow** runs on these files, you can figure out where the problem is. In problem cases where you want to manually tweak the image generation process along the way, you can keep the intermediate files with the **−k** option, modify them appropriately with an image editor, then recombine them with the steps used by the code in **ppmshadow**. See the **ppmshadow.doc** document for additional details and examples of the intermediate files.

Shadows are by default black, as cast by opaque material in the image occluding white light. Use the **−t** option to simulate translucent material, where the shadow takes on the colour of the object that casts it. If the contrast between the image and background is insufficient, the **−t** option may yield unattractive results which resemble simple blurring of the original image.

Because Netpbm used to have a maximum maxval of 255, which meant that the largest convolution kernel **pnmconvol** could use was 11 by 11, **ppmshadow** includes a horrid, CPU-time-burning kludge which, if a blur of greater than 11 is requested, performs an initial convolution with an 11×11 kernel, then calls **pnmsmooth** (which is actually a script that calls pnmconvol with a 3×3 kernel) as many times as the requested blur exceeds 11. It's ugly, but it gets the job done on those rare occasions where you need a blur greater than 11.

If you wish to generate an image at high resolution, then scale it to publication size with **pnmscale** in order to eliminate jagged edges by resampling, it's best to generate the shadow in the original high resolution image, prior to scaling it down in size. If you scale first and then add the shadow, you'll get an unsightly jagged stripe between the edge of material and its shadow, due to resampled pixels intermediate between the image and background obscuring the shadow.

EXIT STATUS

ppmshadow returns status 0 if processing was completed without errors, and a nonzero Unix error code if an error prevented generation of output. Some errors may result in the script aborting, usually displaying error messages from various Netpbm components it uses, without returning a nonzero error code. When this happens, the output file will be empty, so be sure to test this if you need to know if the program succeeded.

SEE ALSO

pnm(5), **pnmmargin**(1), **pnmconvol**(1), **pnmscale**(1), **pnmsmooth**(1), **ppm**(5)

AUTHOR

John Walker <http://www.fourmilab.ch> August 8, 1997

COPYRIGHT

NAME

ppmshift - shift lines of a portable pixmap left or right by a random amount

SYNOPSIS

ppmshift *shift* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Shifts every row of image data to the left or right by a certain amount. The 'shift' parameter determines by how many pixels a row is to be shifted at most.

Another one of those effects I intended to use for MPEG tests. Unfortunately, this program will not help me here - it creates too random patterns to be used for animations. Still, it might give interesting results on still images.

EXAMPLE

Check this out: Save your favourite model's picture from something like alt.binaries.pictures.supermodels (ok, or from any other picture source), convert it to ppm, and process it e.g. like this, assuming the picture is 800x600 pixels:

```
 # take the upper half, and leave it like it is
 pnmcut 0 0 800 300 cs.ppm >upper.ppm

 # take the lower half, flip it upside down, dim it and distort it a little
 pnmcut 0 300 800 300 cs.ppm | pnmflip -tb | ppmdim 0.7 |
    ppmshift 10 >lower.ppm

 # and concatenate the two pieces
 pnmcat -tb upper.ppm lower.ppm >newpic.ppm
```

The resulting picture looks like the image being reflected on a water surface with slight ripples.

SEE ALSO

ppm(5), pnmcut(1), pnmflip(1), ppmdim(1), pnmcat(1)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

ppmspread - displace a portable pixmap's pixels by a random amount

SYNOPSIS

ppmspread *amount* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Moves every pixel around a bit relative to its original position. amount determines by how many pixels a pixel is to be moved around at most.

Pictures processed with this filter will seem to be somewhat dissolved or unfocussed (although they appear more coarse than images processed by something like *pnmconvol*).

SEE ALSO

ppm(5), pnmconvol(1)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

ppmtoacad - convert portable pixmap to AutoCAD database or slide

SYNOPSIS

ppmtoacad [**-dxb**] [**-poly**] [**-background** *colour*] [**-white**] [**-aspect** *ratio*] [**-8**] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces an AutoCAD® slide file or binary database import (.dxb) file as output. If no *ppmfile* is specified, input is read from standard input.

OPTIONS

-dxb An AutoCAD binary database import (.dxb) file is written. This file is read with the DXBIN command and, once loaded, becomes part of the AutoCAD geometrical database and can be viewed and edited like any other object. Each sequence of identical pixels becomes a separate object in the database; this can result in very large AutoCAD drawing files. However, if you want to trace over a bitmap, it lets you zoom and pan around the bitmap as you wish.

-poly If the **-dxb** option is not specified, the output of **ppmtoacad** is an AutoCAD slide file. Normally each row of pixels is represented by an AutoCAD line entity. If **-poly** is selected, the pixels are rendered as filled polygons. If the slide is viewed on a display with higher resolution than the source pixmap, this will cause the pixels to expand instead of appearing as discrete lines against the screen background colour. Regrettably, this representation yields slide files which occupy more disc space and take longer to display.

-background *colour*
Most AutoCAD display drivers can be configured to use any available colour as the screen background. Some users perfer a black screen background, others white, while splinter groups advocate burnt ocher, tawny puce, and shocking grey. Discarding pixels whose closest AutoCAD colour representation is equal to the background colour can substantially reduce the size of the AutoCAD database or slide file needed to represent a bitmap. If no **-background** colour is specified, the screen background colour is assumed to be black. Any AutoCAD colour number may be specified as the screen background; colour numbers are assumed to specify the hues defined in the standard AutoCAD 256 colour palette.

-white Since many AutoCAD users choose a white screen background, this option is provided as a short-cut. Specifying **-white** is identical in effect to **-background 7**.

-aspect *ratio*
If the source pixmap had non-square pixels, the ratio of the pixel width to pixel height should be specified as *ratio*. The resulting slide or .dxb file will be corrected so that pixels on the AutoCAD screen will be square. For example, to correct an image made for a 320x200 VGA/MCGA screen, specify **-aspect 0.8333**.

-8 Restricts the colours in the output file to the 8 RGB shades.

All flags can be abbreviated to their shortest unique prefix.

BUGS

AutoCAD has a fixed palette of 256 colours, distributed along the hue, lightness, and saturation axes. Pixmaps which contain many nearly-identical colours, or colours not closely approximated by AutoCAD's palette, may be poorly rendered.

ppmtoacad works best if the system displaying its output supports the full 256 colour AutoCAD palette. Monochrome, 8 colour, and 16 colour configurations will produce less than optimal results.

When creating a .dxb file or a slide file with the **-poly** option, **ppmtoacad** finds both vertical and horizontal runs of identical pixels and consolidates them into rectangular regions to reduce the size of the output file. This is effective for images with large areas of constant colour but it's no substitute for true raster to vector conversion. In particular, thin diagonal lines are not optimised at all by this process.

Output files can be huge.

SEE ALSO

AutoCAD Reference Manual: *Slide File Format* and *Binary Drawing Interchange (DXB) Files*, **ppm**(5)

AUTHOR

John Walker
Autodesk SA
Avenue des Champs-Montants 14b
CH-2074 MARIN
Suisse/Schweiz/Svizzera/Svizra/Switzerland

Usenet:	kelvin@Autodesk.com
Fax:	038/33 88 15
Voice:	038/33 76 33

NAME

ppmtobmp − convert a portable pixmap into a BMP file

SYNOPSIS

ppmtobmp [**−windows**] [**−os2**] [**−bpp=***bits_per_pixel*] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Microsoft Windows or OS/2 BMP file as output.

OPTIONS

All options can be abbreviated to their shortest unique prefix and you can use a double dash in place of the single dash, GNU-style.

−windows
Tells the program to produce a Microsoft Windows BMP file. (This is the default.)

−os2 Tells the program to produce an OS/2 BMP file. (Before August 2000, this was the default).

−bpp This determines how many bits per pixel you want the BMP file to contain. Only 1, 4, 8, and 24 are possible. By default, **ppmtobmp** chooses the smallest number with which it can represent all the colors in the input image. If you specify a number too small to represent all the colors in the input image, **ppmtobmp** tells you and terminates. You can use **ppmquant** or **ppmdither** to reduce the number of colors in the image.

NOTES

To get a faithful reproduction of the input image, the maxval of the input image must be 255. If it is something else, **ppmtobmp** the colors in the BMP file may be slightly different from the colors in the input.

Windows icons are not BMP files. Use **ppmtowinicon** to create those.

SEE ALSO

bmptoppm(1), **ppmtowinicon**(1), **ppmquant**(1), **ppmdither**(1), **ppm**(5)

AUTHOR

Copyright (C) 1992 by David W. Sanderson.

NAME

ppmtoeyuv - convert a portable pixmap into a Berkeley YUV file

SYNOPSIS

ppmtoeyuv [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Berkeley Encoder YUV (not the same as Abekas YUV) file on the Standard Output file.

With no argument, takes input from Standard Input. Otherwise, *ppmfile* is the file specification of the input file.

ppmtoeyuv handles multi-image PPM input streams, outputting consecutive eyuv images. There must be at least one image, though.

SEE ALSO

eyuvtoppm(1), **ppmtoyuv**(1), **ppm**(5)

NAME

ppmtogif - convert a portable pixmap into a GIF fi le

SYNOPSIS

ppmtogif [**-interlace**] [**-sort**] [**-map** *mapfi le*]
[**−transparent** [=]*color*] [**−alpha** *pgmfi le*] [**−comment** *text*] [**−nolzw**]
[*ppmfi le*]

All options can be abbreviated to their shortest unique prefi x. You may use two hyphens instead of one to designate an option. You may use either white space or equals signs between an option name and its value.

DESCRIPTION

Reads a portable pixmap as input. Produces a GIF fi le as output.

This program creates only individual GIF images. To combine multiple GIF images into an animated GIF, use **gifsicle** (not part of the Netpbm package).

ppmtogif creates either an original GIF87 format GIF fi le or the newer GIF89 format. It creates GIF89 when you request features that were new with GIF89, to wit the **-transparent** or **-comment** options. Otherwise, it creates GIF87. Really old GIF readers conceivably could not recognize GIF89.

OPTIONS

-interlace
Produce an interlaced GIF fi le.

-sort Produces a GIF fi le with a sorted color map.

-map *mapfi le*

Uses the colors found in the *mapfi le* to create the colormap in the GIF fi le, instead of the colors from *ppmfi le.* The *mapfi le* can be any *ppm* fi le; all that matters is the colors in it. If the colors in *ppmfi le* do not match those in *mapfi le* , they are matched to a "best match." A (much) better result can be obtained by using the following fi lter in advance:

ppmquant -fbyd -map *mapfi le*

−transparent *color*
ppmtogif marks the specifi ed color as transparent in the GIF image.

If you don’t specify **-transparent**, **ppmtogif** does not mark any color transparent (except as indicated by the **-alpha** option).

You specify the color as in **ppmmake**(1).**E.g. red** or **rgb:ff/00/0d**. If the color you specify is not present in the image, **ppmtogif** selects instead the color in the image that is closest to the one you specify. Closeness is measured as a cartesian distance between colors in RGB space. If multiple colors are equidistant, **ppmtogif** chooses one of them arbitrarily.

However, if you prefi x your color specifi cation with "=", e.g.

-transparent==red

Only the exact color you specify will be transparent. If that color does not appear in the image, there will be no transparency. **ppmtogif** issues an information message when this is the case.

You cannot specify both **-transparent** and **-alpha**.

-alpha= *pgmfi le*
This option names a PGM fi le that contains an alpha mask for the image. **ppmtogif** Creates fully transparent pixels wherever the alpha mask indicates transparency greater than 50%. The color of those pixels is that specifi ed by the **-alphacolor** option, or black by default.

To do this, **ppmtogif** creates an entry in the GIF colormap in addition to the entries for colors that are actually in the image. It marks that colormap entry as transparent and uses that colormap index in the output image to create a transparent pixel.

The alpha image must be the same dimensions as the input image, but may have any maxval. White means opaque and black means transparent.

You cannot specify both **-transparent** and **-alpha**.

-alphacolor
See **-alpha**.

−comment *text*
Include a comment in the GIF output with comment text *text*. Without this option, there are no comments in the output.

−nolzw This option causes the GIF output, and thus **ppmtogif**, not to use LZW (Lempel-Ziv) compression. As a result, the image fi le is larger and no royalties are owed to the holder of the patent on LZW. See the section LICENSE below.

LZW is a method for combining the information from multiple pixels into a single GIF code. With the **−nolzw** option, **ppmtogif** creates one GIF code per pixel, so it is not doing any compression and not using LZW. However, any GIF decoder, whether it uses an LZW decompressor or not, will correctly decode this uncompressed format. An LZW decompressor would see this as a particular case of LZW compression.

Note that if someone uses an LZW decompressor such as the one in **giftopnm** or pretty much any graphics display program to process the output of **ppmtogif -nolzw** he is then using the LZW patent. But the patent holder has expressed far less interest in enforcing the patent on decoding than on encoding.

SEE ALSO

giftopnm(1), **ppmquant**(1), **pngtopnm**(1), **gifsicle**(1) <http://www.lcdf.org/gifsicle>, **ppm**(5).

AUTHOR

Based on GIFENCOD by David Rowley <mgardi@watdcsu.waterloo.edu>. Lempel-Ziv compression based on "compress".

The non-LZW format is generated by code based on **djpeg** by the Independent Jpeg Group.

Copyright (C) 1989 by Jef Poskanzer.

LICENSE

If you use **ppmtogif** without the **-nolzw** option, you are using a patent on the LZW compression method which is owned by Unisys, and in all probability you do not have a license from Unisys to do so. Unisys typically asks $5000 for a license for trivial use of the patent. Unisys has never enforced the patent against trivial users. The patent expires in 2003.

Rumor has it that IBM also owns a patent covering **ppmtogif**.

A replacement for the GIF format that does not require any patents to use is the PNG format.

NAME

ppmtoicr - convert a portable pixmap into NCSA ICR format

SYNOPSIS

ppmtoicr [**-windowname** *name*] [**-expand** *expand*] [**-display** *display*] [**-rle**] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap file as input. Produces an NCSA Telnet Interactive Color Raster graphic file as output. If *ppmfile* is not supplied, *ppmtoicr* will read from standard input.

Interactive Color Raster (ICR) is a protocol for displaying raster graphics on workstation screens. The protocol is implemented in NCSA Telnet for the Macintosh version 2.3. The ICR protocol shares characteristics of the Tektronix graphics terminal emulation protocol. For example, escape sequences are used to control the display.

ppmtoicr will output the appropriate sequences to create a window of the dimensions of the input pixmap, create a colormap of up to 256 colors on the display, then load the picture data into the window.

Note that there is no icrtoppm tool - this transformation is one way.

OPTIONS

-windowname*name*
Output will be displayed in *name* (Default is to use *ppmfile* or "untitled" if standard input is read.)

-expand*expand* Output will be expanded on display by factor *expand* (For example, a value of 2 will cause four pixels to be displayed for every input pixel.)

-display*display* Output will be displayed on screen numbered *display*

-rle Use run-length encoded format for display. (This will nearly always result in a quicker display, but may skew the colormap.)

EXAMPLES

To display a *ppm* file using the protocol:

```
ppmtoicr ppmfile
```

This will create a window named *ppmfile* on the display with the correct dimensions for *ppmfile,* create and download a colormap of up to 256 colors, and download the picture into the window. The same effect may be achieved by the following sequence:

```
ppmtoicr ppmfile > filename
cat filename
```

To display a GIF file using the protocol in a window titled after the input file, zoom the displayed image by a factor of 2, and run-length encode the data:

```
giftopnm giffile | ppmtoicr -w giffile -r -e 2
```

BUGS

The protocol uses frequent *fflush* calls to speed up display. If the output is saved to a file for later display via *cat,* drawing will be much slower. In either case, increasing the Blocksize limit on the display will speed up transmission substantially.

SEE ALSO

ppm(5)

NCSA Telnet for the Macintosh, University of Illinois at Urbana-Champaign (1989)

AUTHOR

Copyright (C) 1990 by Kanthan Pillay (svpillay@Princeton.EDU), Princeton University Computing and Information Technology.

NAME

ppmtoilbm - convert a portable pixmap into an ILBM file

SYNOPSIS

ppmtoilbm [**-maxplanes**|**-mp** *N*] [**-fixplanes**|**-fp** *N*] [**-ham6**|**-ham8**] [**-dcbits**|**-dcplanes**rgb**]** [**-normal**|**-hamif**|**-hamforce**|**-24if**|**-24force**| -dcif|-dcforce|-cmaponly**]** [**-ecs**|**-aga**] [**-compress**|**-nocompress**] [**-cmethod** *type*] [**-map**ppmfile**]** [**-savemem**] [**ppmfile**]

DESCRIPTION

Reads a portable pixmap as input. Produces an ILBM file as output. Supported ILBM types are:

Normal ILBMs with 1-16 planes.

Amiga HAM with 3-16 planes.

24 bit.

Color map (BMHD + CMAP chunk only, nPlanes = 0).

Unofficial direct color.
1-16 planes for each color component.

Chunks written:
BMHD, CMAP, CAMG (only for HAM), BODY (not for colormap files) unofficial DCOL chunk for direct color ILBM

OPTIONS

Options marked with (*) can be prefixed with a "no", e.g. "-nohamif". All options can be abbreviated to their shortest unique prefix.

-maxplanes | -mp n
(default 5, minimum 1, maximum 16) Maximum planes to write in a normal ILBM. If the pixmap does not fit into <n> planes, ppmtoilbm writes a HAM file (if -hamif is used), a 24bit file (if -24if is used) or a direct color file (if -dcif is used) or aborts with an error.

-fixplanes | -fp n
(min 1, max 16) If a normal ILBM is written, it will have exactly <n> planes.

-hambits | -hamplanes n
(default 6, min 3, max 16) Select number of planes for HAM picture. The current Amiga hardware supports 6 and 8 planes, so for now you should only use this values.

-normal (default)
Turns off -hamif/-24if/-dcif, -hamforce/-24force/-dcforce and -cmaponly. Also sets compression type to byterun1.

-hamif (*)

-24if (*)

-dcif (*)
Write a HAM/24bit/direct color file if the pixmap does not fit into <maxplanes> planes.

-hamforce (*)

-24force (*)

-dcforce (*)
Write a HAM/24bit/direct color file.

-dcbits | -dcplanes r g b
(default 5, min 1, max 16). Select number of bits for red, green & blue in a direct color ILBM.

-ecs (default)
Shortcut for: -hamplanes 6 -maxplanes 5

-aga

Shortcut for: -hamplanes 8 -maxplanes 8

-ham6

Shortcut for: -hamplanes 6 -hamforce

-ham8 Shortcut for: -hamplanes 8 -hamforce

-compress (*) (default)

-cmethod none|byterun1
Compress the BODY chunk. The default compression method is byterun1. Compression requires building the ILBM image in memory; turning compression off allows stream-writing of the image, but the resulting file will usually be 30% to 50% larger. Another alternative is the -savemem option, this will keep memory requirements for compression at a minimum, but is very slow.

-map ppmfile
Write a normal ILBM using the colors in <ppmfile> as the colormap. The colormap file also determines the number of planes, a -maxplanes or -fixplanes option is ignored.

-cmaponly
Write a colormap file: only BMHD and CMAP chunks, no BODY chunk, nPlanes = 0.

-savemem
See the -compress option.

BUGS

HAM pictures will always get a grayscale colormap; a real color selection algorithm might give better results. On the other hand, this allows row-by-row operation on HAM images, and all HAM images of the same depth (no. of planes) share a common colormap, which is useful for building HAM animations.

REFERENCES

Amiga ROM Kernel Reference Manual - Devices (3rd Ed.)
Addison Wesley, ISBN 0-201-56775-X

SEE ALSO

ppm(5), ilbmtoppm(1)

AUTHORS

Copyright (C) 1989 by Jef Poskanzer.
Modified October 1993 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)

NAME

ppmtoleaf - convert PPM image to Interleaf image format

SYNOPSIS

ppmtoleaf [*ppmfi le*]

DESCRIPTION

Reads an Interleaf image fi le as input. Generates a portable pixmap (PPM fi le) as output.

Interleaf is a now-defunct (actually purchased ca. 2000 by BroadVision) technical publishing software company.

SEE ALSO

ppm(5), **ppmquant**(1)

AUTHOR

The program is copyright (C) 1994 by Bill O’Donnell.

NAME

ppmtolj - convert a portable pixmap to an HP LaserJet PCL 5 Color file

SYNOPSIS

ppmtolj [-gamma *val*] [**-resolution 75|100|150|300|600**] [**-delta**] [**-fbat] [**-noreset] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input and converts it into a color file suitable to be printed by an HP color PCL 5 printer.

OPTIONS

-delta Apply delta row compression to reduce the size of the pcl file.

-gamma *int* Gamma correct the image using the integet parameter as a gamma (default 0).

-fbat Suppresses positioning information. The default is to write the sequence ESC&l0E to the output.

-noreset Prevents writing of the reset sequence to the begining and end of the output.

-resolution Set the required output resolution 75|100|150|300|600

REFERENCES

HP PCL 5 & Color Reference Guide

BUGS

None known.

AUTHOR

Copyright (C) 2000 by Jonathan Melvin.(jonathan.melvin@heywood.co.uk)

NAME

ppmtomap - extract all colors from a portable pixmap

DESCRIPTION

This program exists only for backward compatibility.

Use **pnmcolormap**, which replaced it in January 2002.

One trivial difference between **ppmtomap** and **pnmcolormap all** is that if the input is PBM or PGM, **ppmtomap** would produce PPM output, whereas **pnmcolormap all** produces the same kind of output as the input. This should not be very noticeable, though, as PBM and PGM images are usually usable anywhere a PPM image is.

SEE ALSO

pnmcolormap(1).TH**ppmtomitsu**1**29 Jan 1992**

NAME

ppmtomitsu - convert a portable pixmap to a Mitsubishi S340-10 fi le

SYNOPSIS

ppmtomitsu [-sharpness *val*] [**-enlarge** *val*] [**-media** *string*] [**-copy** *val*] [**-dpi300**] [**-tiny**] [*ppmfi le*]

DESCRIPTION

Reads a portable pixmap as input and converts it into a format suitable to be printed by a Mitsubishi S340-10 printer, or any other Mitsubishi color sublimation printer.

The Mitsubishi S340-10 Color Sublimation printer supports 24bit color. Images of the available sizes take so long to transfer that there is a fast method, employing a lookuptable, that ppmtomitsu will use if there is a maximum of 256 colors in the pixmap. ppmtomitsu will try to position your image to the center of the paper, and will rotate your image for you if xsize is larger than ysize. If your image is larger than the media allows, ppmtomitsu will quit with an error message. (We decided that the media were too expensive to have careless users produce misprints.) Once data transmission has started, the job can't be stopped in a sane way without resetting the printer. The printer understands putting together images in the printers memory; ppmtomitsu doesn't utilize this as pnmcat etc provide the same functionality and let you view the result on-screen, too. The S340-10 is the lowest common denominator printer; for higher resolution printers there's the dpi300 option. The other printers also support higher values for enlarge eg, but I don't think that's essential enough to warrant a change in the program.

-sharpness *1-4*
'sharpness' designation. Default is to use the current sharpness.

-enlarge *1-3*
Enlarge by a factor; Default is 1 (no enlarge)

-media *A, A4, AS, A4S*
Designate the media you're using. Default is 1184 x 1350, which will fi t on any media. A is 1216 x 1350, A4 is 1184 x 1452, AS is 1216 x 1650 and A4S is 1184 x 1754. A warning: If you specify a different media than the printer currently has, the printer will wait until you put in the correct media or switch it off.

-copy *1-9*
The number of copies to produce. Default is 1.

-dpi300
Double the number of allowed pixels for a S3600-30 Printer in S340-10 compatibility mode. (The S3600-30 has 300 dpi).

-tiny Memory-safi ng, but always slow. The printer will get the data line-by-line in 24bit. It's probably a good idea to use this if your machine starts paging a lot without this option.

REFERENCES

Mitsubishi Sublimation Full Color Printer S340-10 Specifi cations of Parallel Interface LSP-F0232F

SEE ALSO

ppmquant(1), pnmscale(1), ppm(5)

BUGS

We didn’t find any - yet. (Besides, they’re called features anyway :-) If you should find one, my email-adress is below.

AUTHOR

Copyright (C) 1992, 93 by S.Petra Zeidler, MPIfR Bonn, Germany. (spz@specklec.mpifr-bonn.mpg.de)

NAME

ppmtompeg − encodes MPEG-1 bitstreams

SYNOPSIS

ppmtompeg [*options*] *parameter-file*

DESCRIPTION

ppmtompeg produces an MPEG-1 video stream. param_file is a parameter file which includes a list of input files and other parameters. The file is described in detail below. The -gop, -combine_gops, -frames, and -combine_frames options are all exclusive. This man page is probably incomplete. For complete usage, see the User’s Guide.

OPTIONS

-stat stat_file : causes the encoder to append the statistics to the file *stat_file*. In any case, the statistics are output to stdout. The statistics use the following abbreviations: bits per block (bpb), bits per frame (bpf), seconds per frame (spf), and bits per second (bps).

-quiet num_seconds : causes the program to not report remaining time for at least num_seconds seconds. A negative values tells the program not to report at all. 0 is the default (reports once after each frame). Note that the time remaining is an estimate and does not take into account time to read in frames.

-realquiet : causes the encoder to run silently, with the only screen output being errors. Particularly useful when reading input from stdin.

-no_frame_summary : prevents the program from printing a summary line for each frame

-fbat_dct : forces the encoder to use a more accurate, yet more computationally expensive version of the DCT.

-gop gop_num : causes the encoder to only encode the numbered GOP (first GOP is 0). The parameter file is the same as for normal usage. The output file will be the normal output file with the suffix ".gop.<gop_num>" No sequence info is output.

-combine_gops : causes the encoder to simply combine some GOP files into a single MPEG stream. A sequence header/ender are inserted. In this case, the parameter file need only contain the YUV_SIZE value, an output file, and perhaps a list of input GOP files (see below).

-frames first_frame last_frame : causes the encoder to only encode the frames from first_frame to last_frame, inclusive. The parameter file is the same as for normal usage. The output will be placed in separate files, one per frame, with the file names being the normal output file with the suffix ".frame.<frame num>" No GOP header information is output. (Thus, the parameter file need not include the GOP_SIZE value)

-combine_frames : causes the encoder to simply combine some frames into a single MPEG stream. Sequence and GOP headers are inserted appropriately. In this case, the parameter file need only contain the YUV_SIZE value, the GOP_SIZE value, an output file, and perhaps a list of frame files (see below).

-nice : causes the program to run any remote processes ’nicely.’ This is only relevant if the program is using parallel encoding. (see ’man nice.’)

-max_machines num_machines : causes the program to use no more than num_machines machines as slaves for use in parallel encoding.

-snr : print the signal-to-noise ratio. Prints SNR (Y U V) and peak SNR (Y U V) for each frame. In summary, prints averages of luminance only (Y). SNR is defined as 10*log(variance of original/variance of error). Peak SNR is defined as 20*log(255/RMSE). Note that the encoder will run a little slower if you want it to print the SNR.

-mse : computes the mean squared error per block. Also automatically computes the quality of the images when set, so there is no need to specify -snr then.

-bit_rate_info rate_file : prints bit rate information into the file rate_file. Bit rate info is bits per frame, and also bits per I-frame-to-I-frame.

-mv-histogram : prints histogram of motion vectors as part of statistics. There are three histograms -- one for P, forward B, and backward B vectors. Each histogram is a 2-dimensional array, and there is one entry for each vector in the search window.

PARAMETER FILE

The parameter file MUST contain the following lines (except when using the -combine_gops or -combine_frames options):

PATTERN <pattern>

OUTPUT <output file>

INPUT_DIR <directory>

all input files must reside in this directory. If you want to refer to the current directory, use '.' (an empty INPUT_DIR value would refer to the root directory). If input files will be coming in from standard input, use 'stdin'.

INPUT

This line must be followed by a list of the input files (in display order) and then the line

END_INPUT

There are three types of lines between INPUT and END_INPUT. First, a line may simply be the name of an input file. Secondly, the line may be of the form

<single_star_expr> [x-y]

single_star_expr can have a single '*' in it. It is replaced by all the numbers between x and y inclusive. So, for example, the line

tennis*.ppm [12-15]

is replaced by tennis12.ppm, tennis13.ppm, tennis14.ppm, tennis15.ppm. Uniform zero-padding occurs, as well. For example, the line

football.*.ppm [001-130]

is replaced by football.001.ppm, football.002.ppm, ..., football.009.ppm, football.010.ppm, ..., football.130.ppm. The third type of line is:

<single_star_expr> [x-y+s]

Where the line is treated exactly as above, except that we skip by s. Thus, the line

football.*.ppm [001-130+4]

is replaced by football.001.ppm, football.005.ppm, football.009.ppm, football.013.ppm, etc.

BASE_FILE_FORMAT <YUV or PPM or PNM or JPEG or JMOVIE>

All the input files must be converted to YUV, JPEG(v4), JMOVIE, PNM, or PPM format. This line specifies which of the three formats (actually PPM is a subset of PNM). The reason for having a separate PPM option is for simplicity. If your files are RAWBITS ppm files, then use the PPM option rather than the PNM. Also, depending on the system, file reads will go much faster with the PPM option (as opposed to PNM).

INPUT_CONVERT <conversion command>

You must specify how to convert a file to the base file format. In the conversion command, each '*' is replaced by the filename (the items listed between INPUT and END_INPUT). If no conversion is necessary, then you would just say:

INPUT_CONVERT *

If you had a bunch of gif files, you might say:

INPUT_CONVERT giftoppm *

If you have a bunch of separate a.Y, a.U, and a.V files, then you might say:

INPUT_CONVERT cat *.Y *.U *.V

Input conversion is not allowed with input from stdin.

GOP_SIZE <n>

n is roughly the number of frames in a Group of Pictures (roughly because a GOP must begin with an I-frame)

SLICES_PER_FRAME <n>

n is roughly the number of slices per frame. Note, at least one MPEG player may complain if slices do not start at the left side of an image. To ensure this does not happen, make sure the number of rows is divisible by SLICES_PER_FRAME.

PIXEL <FULL or HALF>
: use half-pixel motion vectors, or only full-pixel ones

RANGE <n>
: use a search range of +/- n pixels

PSEARCH_ALG <algorithm>
: algorithm must be one of {EXHAUSTIVE, TWOLEVEL, SUBSAMPLE, LOGARITHMIC}. Tells what kind of search procedure should be used for P-frames. Exhaustive gives the best compression, but logarithmic is the fastest. You select the desired combination of speed and compression. TWOLEVEL is an exhaustive full-pixel search, followed by a local half-pixel search around the best full-pixel vector (the PIXEL option is ignored for this search algorithm).

BSEARCH_ALG <algorithm>
: algorithm must be one of {SIMPLE, CROSS2, EXHAUSTIVE}. Tells what kind of search procedure should be used for B-frames. Simple means find best forward and backward vectors, then interpolate. Cross2 means find those two vectors, then see what backward vector best matches the best forward vector, and vice versa. Exhaustive does an n-squared search and is EXTREMELY slow in relation to the others (Cross2 is about twice as slow as Simple).

IQSCALE <n>
: use n as the qscale for I-frames

PQSCALE <n>
: use n as the qscale for P-frames

BQSCALE <n>
: use n as the qscale for B-frames

REFERENCE_FRAME <ORIGINAL or DECODED>
: If ORIGINAL is specified, then the original images are used when computing motion vectors. To be more accurate, use DECODED, in which the decoded images are used. This should increase the quality of the image, but will take a bit longer to encode.

The following lines are optional:

FORCE_I_ALIGN
: This option is only relevant for parallel execution (see below). It forces each processor to encode a block of N frames, where N must be a multiple of the pattern length. Since the first frame in any pattern is an I-frame, this forces each block encoded by a processor to begin with an I-frame.

foo

NOTES

If the BASE_FILE_FORMAT is YUV, then the parameter file must contain:

YUV_SIZE <w>x<h>

where w = width, h = height (in pixels) of image, and

YUV_FORMAT <ABEKAS or PHILLIPS or UCB or EYUV or pattern>.

See the file doc/INPUT.FORMAT for more information.

If the -combine-gops option is used, then only the YUV_SIZE and OUTPUT values need be specified in the parameter file. In addition, the parameter file may specify input GOP files in the same manner as normal input files -- except instead of using INPUT_DIR, INPUT, and END_INPUT, use GOP_INPUT_DIR, GOP_INPUT, and GOP_END_INPUT. If no input GOP files are specified, then the default is to use the output file name with suffix ".gop.<gop_num>" starting from 0 as the input files.

If the -combine-frames option is used, then only the YUV_SIZE, GOP_SIZE, and OUTPUT values need be specified in the parameter file. In addition, the parameter file may specify input frame files in the same manner as normal input files -- except instead of using INPUT_DIR, INPUT, and

END_INPUT, use FRAME_INPUT_DIR, FRAME_INPUT, and FRAME_END_INPUT. If no input frame files are specified, then the default is to use the output file name with suffix ".frame.<frame_num>" starting from 0 as the input files.

Any number of spaces and tabs may come between each option and value. Lines beginning with '#' are ignored. Any other lines are ignored except for those between INPUT and END_INPUT. This allows you to use the same parameter file for normal usage and for -combine_gops and -combine_frames.

The encoder is case-sensitive so, except for file names and directories, everything should be in upper case.

The lines may appear in any order, except the following exceptions. INPUT must appear before END_INPUT (also, GOP_INPUT before GOP_END_INPUT and FRAME_INPUT before FRAME_END_INPUT). All lines between INPUT and END_INPUT must be the frames in play order.

The encoder is prepared to handle up to 16 B frames between reference frames when encoding with input from stdin. To increase this amount, change the constant B_FRAME_RUN in frame.c and recompile.

PARALLEL OPERATION

The encoder may be run on multiple machines at once. To do so, add a line "PARALLEL" in the parameter file, followed by a listing, one machine per line, then "END_PARALLEL". Each of the lines should be in one of two forms. If the machine has access to the file server, then the line should be:

```
<machine> <user> <executable>
```

The executable is normally ppmtompeg (you may need to give the complete path if you've built for different architectures). If the machine is a remote machine, then the line should be:

```
REMOTE <machine> <user> <executable> <parameter file>
```

Full paths should generally be used when describing executables and parameter files. This INCLUDES the parameter file given as an argument to the original call to ppmtompeg. Also, .rhosts files on the appropriate machines should have the appropriate information.

The encoder will use the original machine for the master and I/O server processes, and uses the listed machines as slaves to do the computation.

Optional lines are

RSH <remote shell command>
: The encoder uses the remote shell command to start processes on other machines. The default command is 'rsh.' If your machine supports a different command, specify it here.

PARALLEL_TEST_FRAMES <n>
: n is the number of frames to encode initially on each processor

PARALLEL_TIME_CHUNKS <t>
: subsequently, each slave processor will be asked to encode for approximately t seconds. Smaller values of <t> increase communication, but improve load balancing.

: The default values for these two options are n = 3 frames and t = 30 seconds.

PARALLEL_PERFECT
: If this line is present, then scheduling is done on the assumption that work distribution will be perfectly even -- meaning that each machine is about the same speed. The frames will simply be divided up evenly between the processors. This has the advantage of very minimal scheduling overhead, but is obviously wrong if machines have varying speeds, or if the network load makes performance uneven.

VERSION

This is version 1.5 it contins new features and bug fixes from version 1.3.

BUGS

No known bugs, but if you find any, report them to mpeg-bugs@plateau.cs.berkeley.edu.

AUTHORS

Kevin Gong - University of California, Berkeley, keving@cs.berkeley.edu

Ketan Patel - University of California, Berkeley, kpatel@cs.berkeley.edu

Dan Wallach - University of California, Berkeley, dwallach@cs.berkeley.edu

Darryl Brown - University of California, Berkeley, darryl@cs.berkeley.edu

Eugene Hung - University of California, Berkeley, eyhung@cs.berkeley.edu

Steve Smoot - University of California, Berkeley, smoot@cs.berkeley.edu

NAME

ppmtoneo - convert a portable pixmap into an Atari Neochrome .neo file

SYNOPSIS

ppmtoneo [*ppmfile*]

DESCRIPTION

Reads a PPM image as input. Produces an Atari Neochrome .neo file as output.

SEE ALSO

neotoppm(1), **ppm**(5)

AUTHOR

Copyright (C) 2001 by Teemu Hukkanen <tjhukkan@iki.fi>, based on ppmtopi1 by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

ppmtopcx - convert a portable pixmap into a PCX file

SYNOPSIS

ppmtopcx [**-24bit**] [**-packed**] [**-xpos=***cols*] [**-ypos=***rows*] [*ppmfile*]

DESCRIPTION

Reads a PPM image as input. Produces a PCX file as output. The type of the PCX file depends on the number of colors in the pixmap:

16 colors or less:
: 1 bit/pixel, 1-4 planes.

256 colors or less:
: 8 bits/pixel, 1 plane, colormap at the end of the file.

More than 256 colors:
: 24bit truecolor file (8 bits/pixel, 3 planes).

OPTIONS

-24bit Produce a 24bit truecolor file, even if the pixmap has 256 colors or less.

-packed
: Use "packed pixel" format for files with 16 colors or less: 1, 2, or 4 bits/pixel, 1 plane.

-xpos=*cols*

-ypos=*rows*
: These options set the position of the image in some field (e.g. on a screen) in columns to the right of the left edge and rows below the top edge. The PCX format contains image position information. Don't confuse this with the position of an area of interest within the image. For example, using **pnmpad** to add a 10 pixel left border to an image and then converting that image to PCX with xpos = 0 is not the same as converting the original image to PCX and setting xpos = 10.

  The values may be from -32767 to 32768.

  The default for each is zero.

SEE ALSO

pcxtoppm(1), **ppm**(5)

AUTHORS

Copyright (C) 1994 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)
Based on previous work by Michael Davidson.

NAME

ppmtopgm - convert a portable pixmap into a portable graymap

SYNOPSIS

ppmtopgm [*ppmfi le*]

DESCRIPTION

Reads a portable pixmap as input. Produces a portable graymap as output. The output is a "black and white" rendering of the original image, as in a black and white photograph. The quantization formula used is .299 r + .587 g + .114 b.

Note that although there is a **pgmtoppm** program, it is not necessary for simple conversions from pgm to ppm , because any ppm program can read pgm (and pbm) fi les automatically. **pgmtoppm** is for colorizing a pgm fi le. Also, see **ppmtorgb3** for a different way of converting color to gray. And **ppmdist** generates a grayscale image from a color image, but in a way that makes it easy to differentiate the original colors, not necessarily a way that looks like a black and white photograph.

QUOTE

Cold-hearted orb that rules the night
Removes the colors from our sight
Red is gray, and yellow white
But we decide which is right
And which is a quantization error.

SEE ALSO

pgmtoppm(1),**ppmtorgb3**(1),**rgb3toppm**(1),**ppmdist**(1),**ppm**(5),**pgm**(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

ppmtopi1 - convert a portable pixmap into an Atari Degas .pi1 file

SYNOPSIS

ppmtopi1 [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces an Atari Degas .pi1 file as output.

SEE ALSO

pi1toppm(1), ppm(5), pbmtopi3(1), pi3topbm(1)

AUTHOR

Copyright (C) 1991 by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

ppmtopict - convert a portable pixmap into a Macintosh PICT file

SYNOPSIS

ppmtopict [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Macintosh PICT file as output.

The generated file is only the data fork of a picture. You will need a program such as *mcvert* to generate a Macbinary or a BinHex file that contains the necessary information to identify the file as a PICT file to MacOS.

Even though PICT supports 2 and 4 bits per pixel, *ppmtopict* always generates an 8 bits per pixel file.

BUGS

The picture size field is only correct if the output is to a file since writing into this field requires seeking backwards on a file. However the PICT documentation seems to suggest that this field is not critical anyway since it is only the lower 16 bits of the picture size.

SEE ALSO

picttoppm(1), ppm(5), mcvert(1)

AUTHOR

Copyright (C) 1990 by Ken Yap <ken@cs.rocester.edu>.

NAME

ppmtopj - convert a portable pixmap to an HP PaintJet file

SYNOPSIS

ppmtopj [-gamma *val*] [**-xpos** *val*] [**-ypos** *val*] [**-back dark**|**lite**] [**-rle**] [**-center**] [**-render none**|**snap**|**bw**|**dither**|**diffuse**|**monodither**|**monodiffuse**|**clusterdither**|**monoclusterdither**] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input and converts it into a format suitable to be printed by an HP PaintJet printer.

For best results, the input file should be in 8-color RGB form; i.e. it should have only the 8 binary combinations of full-on and full-off primaries. You could get this by sending the input file through *ppmquant -map* with a map file such as:

```
P3
8 1
255
0 0 0      255 0 0    0 255 0    0 0 255
255 255 0  255 0 255  0 255 255  255 255 255
```

Or else you could use use *ppmdither -red 2 -green 2 -blue 2.*

OPTIONS

-rle	Run length encode the image. (This can result in larger images)
-back	Enhance the foreground by indicating if the background is light or dark compated to the foreground.
-render *alg*	Use an internal rendering algorithm (default dither).
-gamma *int*	Gamma correct the image using the integet parameter as a gamma (default 0).
-center	Center the image to an 8.5 by 11 page
-xpos *pos*	Move by pos pixels in the x direction.
-ypos *pos*	Move by pos pixels in the y direction.

REFERENCES

HP PaintJet XL Color Graphics Printer User's Guide

SEE ALSO

pnmdepth(1), ppmquant(1), ppmdither(1), ppm(5)

BUGS

Most of the options have not been tested because of the price of the paper.

AUTHOR

Copyright (C) 1991 by Christos Zoulas.

NAME

ppmtopjxl - convert a portable pixmap into an HP PaintJet XL PCL file

SYNOPSIS

ppmtopjxl [-nopack] [-gamma <*n*>] [-presentation] [-dark] [-diffuse] [-cluster] [-dither] [-xshift <*s*>] [-yshift <*s*>] [-xshift <*s*>] [-yshift <*s*>] [-xsize|-width|-xscale <*s*>] [-ysize|-height|-yscale <*s*>] [ppmfile]

DESCRIPTION

Reads a portable pixmap as input. Produces a PCL file suitable for printing on an HP PaintJet XL printer as output.

The generated file is not suitable for printing on a normal PrintJet printer. The **−nopack** option generates a file which does not use the normal TIFF 4.0 compression method. This file might be printable on a normal PaintJet printer (not an XL).

The **−gamma** option sets the gamma correction for the image. The useful range for the PaintJet XL is approximately 0.6 to 1.5.

The rendering algorithm used for images can be altered with the **-dither, -cluster,** and **-diffuse** options. These options select ordered dithering, clustered ordered dithering, or error diffusion respectively. The **−dark** option can be used to enhance images with a dark background when they are reduced in size. The **−presentation** option turns on presentation mode, in which two passes are made over the paper to increase ink density. This should be used only for images where quality is critical.

The image can be resized by setting the **−xsize** and **−ysize** options. The parameter to either of these options is interpreted as the number of dots to set the width or height to, but an optional dimension of ‘**pt**’ (points), ‘**dp**’ (decipoints), ‘**in**’ (inches), or ‘**cm**’ (centimetres) may be appended. If only one dimension is specified, the other will be scaled appropriately.

The options **−width** and **−height** are synonyms of **−xsize** and **−ysize.**

The **−xscale** and **−yscale** options can alternatively be used to scale the image by a simple factor.

The image can be shifted on the page by using the **−xshift** and **−yshift** options. These move the image the specified dimensions right and down.

SEE ALSO

ppm(5)

AUTHOR

Angus Duggan

NAME

ppmtopuzz - convert a portable pixmap into an X11 "puzzle" file

SYNOPSIS

ppmtopuzz [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces an X11 "puzzle" file as output. A "puzzle" file is for use with the *puzzle* program included with the X11 distribution - *puzzle*'s **-picture** flag lets you specify an image file.

SEE ALSO

ppm(5), puzzle(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

ppmtorgb3 - separate a portable pixmap into three portable graymaps

SYNOPSIS

ppmtorgb3 [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Writes three portable graymaps as output, one each for red, green, and blue.

The output filenames are constructed by taking the input filename, stripping off any extension, and appending ".red", ".grn", and ".blu". For example, separating lenna.ppm would result in lenna.red, lenna.grn, and lenna.blu. If the input comes from stdin, the names are noname.red, noname.grn, and noname.blu.

SEE ALSO

rgb3toppm(1), ppmtopgm(1), pgmtoppm(1), ppm(5), pgm(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

ppmtosixel - convert a portable pixmap into DEC sixel format

SYNOPSIS

ppmtosixel [**-raw**] [**-margin**] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces sixel commands (SIX) as output. The output is formatted for color printing, e.g. for a DEC LJ250 color inkjet printer.

If RGB values from the PPM file do not have maxval=100, the RGB values are rescaled. A printer control header and a color assignment table begin the SIX file. Image data is written in a compressed format by default. A printer control footer ends the image file.

OPTIONS

-raw If specified, each pixel will be explicitly described in the image file. If **-raw** is not specified, output will default to compressed format in which identical adjacent pixels are replaced by "repeat pixel" commands. A raw file is often an order of magnitude larger than a compressed file and prints much slower.

-margin
If **-margin** is not specified, the image will be start at the left margin (of the window, paper, or whatever). If **-margin** is specified, a 1.5 inch left margin will offset the image.

PRINTING

Generally, sixel files must reach the printer unfiltered. Use the lpr -x option or cat filename > /dev/tty0?.

BUGS

Upon rescaling, truncation of the least significant bits of RGB values may result in poor color conversion. If the original PPM maxval was greater than 100, rescaling also reduces the image depth. While the actual RGB values from the ppm file are more or less retained, the color palette of the LJ250 may not match the colors on your screen. This seems to be a printer limitation.

SEE ALSO

ppm(5)

AUTHOR

Copyright (C) 1991 by Rick Vinci.

NAME

ppmtotga - convert portable pixmap into a TrueVision Targa file

SYNOPSIS

ppmtotga [**-mono**|**-cmap**|**-rgb**] [**-norle**] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a TrueVision Targa file as output.

OPTIONS

-mono Forces Targa file to be of type 8 bit monochrome. Input must be a portable bitmap or a portable graymap.

-cmap Forces Targa file to be of type 24 bit colormapped. Input must be a portable bitmap, a portable graymap or a portable pixmap containing no more than 256 distinct colors.

-rgb Forces Targa file to be of type 24 bit unmapped color.

-norle Disables run-length encoding, in case you have a Targa reader which can't read run-length encoded files.

All flags can be abbreviated to their shortest unique prefix. If no file type is specified the most highly constained compatible type is used, where monochrome is more constained than colormapped which is in turn more constained than unmapped.

BUGS

Does not support all possible Targa file types. Should really be in PNM, not PPM.

SEE ALSO

tgatoppm(1), ppm(5)

AUTHOR

Copyright (C) 1989, 1991 by Mark Shand and Jef Poskanzer.

NAME

ppmtouil - convert a portable pixmap into a Motif UIL icon file

SYNOPSIS

ppmtouil [**-name** *uilname*] [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces a Motif UIL icon file as output.

If the program was compiled with an rgb database specified, and a RGB value from the ppm input matches a RGB value from the database, then the corresponding color name mnemonic is printed in the UIL's colormap. If no rgb database was compiled in, or if the RGB values don't match, then the color will be printed with the #RGB, #RRGGBB, #RRRGGGBBB, or #RRRRGGGGBBBB hexadecimal format.

OPTIONS

-name Allows you to specify the prefix string which is printed in the resulting UIL output. If not specified, will default to the filename (without extension) of the ppmfile argument. If **-name** is not specified and no ppmfile is specified (i.e. piped input), the prefix string will default to the string "noname".

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

ppm(5)

AUTHOR

Converted by Jef Poskanzer from ppmtoxpm.c, which is Copyright (C) 1990 by Mark W. Snitily

NAME

ppmtowinicon − convert 1 or more portable pixmaps into a Windows .ico file

SYNOPSIS

ppmtowinicon [*−andpgms*] [*−output output.ico*] [*ppmfiles...]*

DESCRIPTION

Reads one or more portable pixmaps as input. Produces a Microsoft Windows .ico file as output.

A Windows icon contains 1 or more images, at different resolutions and color depths.

Microsoft recommends including at least the following formats in each icon.

16 x 16 - 4 bpp

32 x 32 - 4 bpp

48 x 48 - 8 bpp

Default I/O is STDIN/STDOUT.

OPTIONS

−andpgms

If this option is given, every other file is read as an alpha mask to be used by windows for transparancy data for the previous image. (These are set to blank by default). The alpha mask is a PGM image, where any value less than or equal to the maxval means transparent, and anything greater than the maxval means opaque. Note that as with all Netpbm programs, you may use a PBM file here and it will be used as if it were the equivalent PGM.

−output output.ico

File to write. By default, the icon is written to stdout.

SEE ALSO

winicontoppm(1), **ppm**(5)

AUTHOR

Copyright (C) 2000 by Lee Benfield.

NAME

ppmtoxpm - convert a PPM iamge into an X11 pixmap

SYNOPSIS

ppmtoxpm [**-name=***xpmname*] [**-rgb=***rgb-textfile*] [**-alphamask=***pgmfile*] [*ppmfile*]

Minimum unique abbrevations are acceptable.

DESCRIPTION

Reads a portable pixmap as input. Produces X11 pixmap (version 3) as output which can be loaded directly by the XPM library.

For example, to convert the file "dot" (found in /usr/include/X11/bitmaps), from xbm to xpm one could specify

```
xbmtopbm dot | ppmtoxpm -name dot
```

or, with a rgb text file (in the local directory)

```
xbmtopbm dot | ppmtoxpm -name dot -rgb rgb.txt
```

OPTIONS

-name=*xpmname*

The **-name** option allows you to specify the prefix string which is printed in the resulting XPM output. If not specified, will default to the filename (without extension) of the <ppmfile> argument. If you do not specify **-name** or *ppmfile,* (i.e. your input is from Standad Input), the prefix string defaults to the string **noname**.

rgb=*rgb-textfile*

The **-rgb** option allows you to specify an X11 rgb text file for the lookup of color name mnemonics. This rgb text file is typically the /usr/lib/X11/rgb.txt of the MIT X11 distribution, but any file using the same format may be used. When specified and a RGB value from the ppm input matches a RGB value from the <rgb-textfile>, then the corresponding color name mnemonic is printed in the XPM's colormap. If you don't specify **-rgb or if the RGB values don't** match, then **ppmtoxpm produces the color specifications in the #RGB, #RRGGBB,** #RRRGGGBBB, or #RRRRGGGGBBBB hexadecimal format.

-alphamask=*pgmfile*

This option names a PGM file to use as an alpha (transparency) mask. The file must contain an image the same dimensions as the input image. **ppmtoxpm** marks as transparent any pixel whose position in the alpha mask image is at most half white.

If you don't specify **-alphamask**, **ppmtoxpm** makes all pixels in the output opaque.

ppmcolormask is one way to generate an alpha mask file. You might also generate it by extracting transparency information from an XPM file with the **-alphaout** option to **xpmtoppm**. There are similar options on other Netpbm converters that convert from formats that include transparency information too.

LIMITATIONS

An option to match the closest (rather than exact) color name mnemonic from the rgb text would be a desirable enhancement.

Truncation of the least significant bits of a RGB value may result in nonexact matches when performing color name mnemonic lookups.

SEE ALSO

ppmcolormask(1), **xpmtoppm**(1), **ppm**(5)
XPM Manual by Arnaud Le Hors lehors@mirsa.inria.fr

AUTHOR

This tool was developed for Schlumberger Technologies, ATE Division, and with their permission is being made available to the public with the above copyright notice and permission notice.

Upgraded to XPM2 by
  Paul Breslaw, Mecasoft SA, Zurich, Switzerland (paul@mecazh.uu.ch)
  Thu Nov 8 16:01:17 1990

Upgraded to XPM version 3 by
  Arnaud Le Hors (lehors@mirsa.inria.fr)
  Tue Apr 9 1991

NAME

ppmtoyuv - convert a portable pixmap into an Abekas YUV file

SYNOPSIS

ppmtoyuv [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces an Abekas YUV file as output.

SEE ALSO

yuvtoppm(1), ppm(5)

AUTHOR

Marc Boucher <marc@PostImage.COM>, based on Example Conversion Program, A60/A64 Digital Video Interface Manual, page 69.

Copyright (C) 1991 by DHD PostImage Inc.

Copyright (C) 1987 by Abekas Video Systems Inc.

NAME

ppmtoyuvsplit - convert a portable pixmap into 3 subsampled raw YUV files

SYNOPSIS

ppmtoyuvsplit *basename* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Produces 3 raw files basename.Y, basename.U and basename.V as output. These files are the subsampled raw YUV representation of the input pixmap, as required by the Stanford MPEG codec. The subsampling is done by arithmetic mean of 4 pixels colors into one. The YUV values are scaled according to CCIR.601, as assumed by MPEG.

SEE ALSO

mpeg(1), ppm(5)

AUTHOR

Copyright (C) 1993 by Andre Beck. (Andre_Beck@IRS.Inf.TU-Dresden.de)

Based on ppmtoyuv.c

NAME

ppmtv - make a portable pixmap look like taken from an American TV

SYNOPSIS

ppmtv *dimfactor* [*ppmfile*]

DESCRIPTION

Reads a portable pixmap as input. Dims every other row of image data down by the specified dim factor. This factor may be in the range of 0.0 (the alternate lines are totally black) to 1.0 (original image).

This creates an effect similar to what I've once seen in the video clip 'You could be mine' by Guns'n'Roses. In the scene I'm talking about you can see John Connor on his motorbike, looking up from the water trench (?) he's standing in. While the camera pulls back, the image gets 'normal' by brightening up the alternate rows of it. I thought this would be an interesting effect to try in MPEG. I did not yet check this out, however. Try for yourself.

SEE ALSO

ppm(5), ppmdim(1)

AUTHOR

Copyright (C) 1993 by Frank Neumann

NAME

psidtopgm - convert PostScript "image" data into a portable graymap

SYNOPSIS

psidtopgm *width height bits/sample* [*imagedata*]

DESCRIPTION

Reads the "image" data from a PostScript file as input. Produces a portable graymap as output.

This program is obsoleted by **pstopnm**. What follows was written before **pstopnm** existed.

This is a very simple and limited program, and is here only because so many people have asked for it. To use it you have to **manually** extract the readhexstring data portion from your PostScript file, and then give the width, height, and bits/sample on the command line. Before you attempt this, you should **at least** read the description of the "image" operator in the PostScript Language Reference Manual.

It would probably not be too hard to write a script that uses this filter to read a specific variety of PostScript image, but the variation is too great to make a general-purpose reader. Unless, of course, you want to write a full-fledged PostScript interpreter...

SEE ALSO

pnmtops(1), pgm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

pstopnm - convert a PostScript file into a portable anymap

SYNOPSIS

pstopnm [**−stdout**] [**−forceplain**] [**−help**] [**−llx** *s*] [**−lly** *s*] [**−landscape**] [**−portrait**] [**−nocrop**] [**−pbm** |**−pgm** |**−ppm**] [**−urx** *s*] [**−ury** *s*] [**−verbose**] [**−xborder** *n*] [**−xmax** *n*] [**−xsize** *f*] [**−yborder** *f*] [**−ymax** *n*] [**−ysize** *n*] *psfile*[**.ps**]

DESCRIPTION

Reads a PostScript file as input. Produces PBM, PGM, or PPM files as output. This program simply uses **GhostScript** to render a PostScript file with its PNM device drivers. If you don't have **GhostScript** installed (invoked by a **gs** command), or the version you have installed was not built with the relevant PNM device drivers, **pstopnm** will fail. You can see if you have the proper environment by issuing the command **gs --help** . If it responds and lists under "Available Devices" **pbm**, **pbmraw**, **pgm**, **pgmraw**, **pnm**, **pnmraw**, **ppm**, or **ppmraw**, you're in business.

pstopnm does not use the Netpbm libraries to generate the output files, so may not be entirely consistent with most Netpbm programs.

psfile[**.ps**] is the name of the input file. **.pstopnm** will add the **ps** to the end of the name you specify if no file exists by the exact name you specify, but one with added does. Use **-** to indicate Standard Input.

If you use the **-stdout** option, **pstopnm** outputs images of all the pages as a multi-image file to Standard Output. Otherwise, **pstopnm** creates one file for each page in the Postscript document. The files are named as follows: If the input file is named **psfile.ps**, the name of the files will be **psfile001.ppm**, **psfile002.ppm**, etc. The filetype suffix is **.ppm**, **.pgm**, or **.pbm**, depending on which kind of output you choose with your invocation options. If the input file name does not end in **.ps**, the whole file name is used in the output file name. For example, if the input file is named **psfile.old**, the output file name is **psfile.old001.ppm**, etc.

Note that the output file selection is inconsistent with most Netpbm programs, because it does not default to Standard Output. This is for historical reasons, based on the fact that the Netpbm formats did not always provide for a sequence of images in a single file.

Each output file contains the image of a rectangular part of the page to which it pertains. The selected area will always be centered in the output file, and may have borders around it. The image area to be extracted from the PostScript file and rendered into a portable anymap is defined by four numbers, the lower left corner and the upper right corner x and y coordinates. These coordinates are usually specified by the BoundingBox comment in the PostScript file header, but they can be overridden by the user by specifying one or more of the following options: **−llx**, **−lly**, **−urx**, and **−ury**. The presence and thickness of a border to be left around the image area is controlled by the use of the options **−xborder** and **−yborder**. If **pstopnm** does not find BoundingBox parameters in the input, and you don't specify image area coordinates on the command line, **pstopnm** uses default values. If your input is from Standard Input, **pstopnm** does not use the BoundingBox parameters (due to the technical difficulty of extracting that information and still feeding the file to Ghostscript), so you either have to specify the image area coordinates or take the default.

Unless you specify both output file width and height, via the **−xsize** and **−ysize** options, **pstopnm** maps the document into the output image by preserving its aspect ratio.

It has been reported that on some Postscript Version 1 input, Ghostscript, and therefore **pstopnm**, produces no output. To solve this problem, you can convert the file to Postscript Version 3 with the program **ps2ps**. It is reported that the program **pstops** does not work.

OPTIONS

−forceplain

forces the output file to be in plain (text) format. Otherwise, it is in raw (binary) format. See **pbm**(1), etc.

−llx *bx* selects *bx* as the lower left corner x coordinate (in inches).

−lly *by* selects *by* as the lower left corner y coordinate (in inches).

−landscape
renders the image in landscape mode.

−portrait
renders the image in portrait mode.

−nocrop
does not crop the output image dimensions to match the PostScript image area dimensions.

−pbm −pgm −ppm
selects the format of the output file. By default, all files are rendered as portable pixmaps (ppm format).

−stdout
causes output to go to Standard Output instead of to regular files, one per page (see description of output files above). Use **pnmsplit** to extract individual pages from Standard Output.

−urx *tx* selects *tx* as the upper right corner x coordinate (in inches).

−ury *ty* selects *ty* as the upper right corner y coordinate (in inches).

−verbose
prints processing information to stdout.

−xborder *frac*
specifies that the border width along the Y axis should be *frac* times the document width as specified by the bounding box comment in the PostScript file header. The default value is 0.1.

−xmax *xs*
specifies that the maximum output image width should have a size less or equal to *xs* pixels (default: 612).

−xsize *xsize*
specifies that the output image width must be exactly *xs* pixels.

−yborder *frac*
specifies that the border width along the X axis should be *frac* times the document width as specified by the bounding box comment in the PostScript file header. The default value is 0.1.

−ymax *ys*
specifies that the maximum output image height should have a size less or equal to *ys* pixels (default: 792).

−ysize *ys*
specifies that the output image height must be exactly *ys* pixels.

BUGS

The program will produce incorrect results with PostScript files that initialize the current transformation matrix. In these cases, page translation and rotation will not have any effect. To render these files, probably the best bet is to use the following options:

```
pstopnm -xborder 0 -yborder 0 -portrait -nocrop file.ps
```

Additional options may be needed if the document is supposed to be rendered on a medium different from letter-size paper.

SEE ALSO

gs(1), **pstofits**(1), **pnmtops**(1), **psidtopgm**(1), **pbmtolps**(1), **pbmtoepsi**(1), **pnmsplit**(1)

COPYRIGHT

Copyright (c) 1992 Smithsonian Astrophysical Observatory
PostScript is a Trademark of Adobe Systems Incorporated.

AUTHOR

Alberto Accomazzi, WIPL, Center for Astrophysics.

NAME

qrttoppm - convert output from the QRT ray tracer into a portable pixmap

SYNOPSIS

qrttoppm [*qrtfi le*]

DESCRIPTION

Reads a QRT fi le as input. Produces a portable pixmap as output.

SEE ALSO

ppm(5)

AUTHOR

Copyright (C) 1989 by Jef Poskanzer.

NAME

rasttopnm - convert a Sun rasterfile into a portable anymap

SYNOPSIS

rasttopnm [*rastfile*]

DESCRIPTION

Reads a Sun rasterfile as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's black & white, a *pbm* file is written, else if it's grayscale a *pgm* file, else a *ppm* file. The program tells you which type it is writing.

SEE ALSO

pnmtorast(1), pnm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

rawtopgm - convert raw grayscale bytes into a portable graymap

SYNOPSIS

rawtopgm [**-bpp** [**1**|**2**]] [**-littleendian**] [**-maxval** *N*] [**-headerskip** *N*] [**-rowskip** *N*] [**-tb**|**-topbottom**] [*width height*] [*imagefi le*]

DESCRIPTION

Reads raw grayscale values as input. Produces a PGM fi le as output. The input fi le is just a sequence of pure binary numbers, either one or two bytes each, either bigendian or littleendian, representing gray values. They may be arranged either top to bottom, left to right or bottom to top, left to right. There may be arbitrary header information at the start of the fi le (to which **rawtopgm** pays no attention at all other than the header's size).

Arguments to **rawtopgm** tell how to interpret the pixels (a function that is served by a header in a regular graphics format).

The *width* and *height* parameters tell the dimensions of the image. If you omit these parameters, **rawtopgm** assumes it is a quadratic image and bases the dimensions on the size of the input stream. If this size is not a perfect square, **rawtopgm** fails.

When you don't specify *width* and *height*, **rawtopgm** reads the entire input stream into storage at once, which may take a lot of storage. Otherwise, **rawtopgm** ordinarily stores only one row at a time.

If you don't specify *imagefi le*, or specify **-**, the input is from Standard Input.

The PGM output is to Standard Output.

OPTIONS

-maxval *N*

N is the maxval for the gray values in the input, and is also the maxval of the PGM output image. The default is the maximum value that can be represented in the number of bytes used for each sample (i.e. 255 or 65535).

-bpp [**1**|**2**]

tells the number of bytes that represent each sample in the input. If the value is **2**, The most signifi cant byte is fi rst in the stream.

The default is 1 byte per sample.

-littleendian

says that the bytes of each input sample are ordered with the least signifi cant byte fi rst. Without this option, **rawtopgm** assumes MSB fi rst. This obviously has no effect when there is only one byte per sample.

-headerskip *N*

rawtopgm skips over *N* bytes at the beginning of the stream and reads the image immediately after. The default is 0.

This is useful when the input is actually some graphics format that has a descriptive header followed by an ordinary raster, and you don't have a program that understands the header or you want to ignore the header.

-rowskip *N*

If there is padding at the ends of the rows, you can skip it with this option. Note that rowskip need not be an integer. Amazingly, I once had an image with 0.376 bytes of padding per row. This turned out to be due to a file-transfer problem, but I was still able to read the image.

Skipping a fractional byte per row means skipping one byte per multiple rows.

-bt -bottomfirst

By default, **rawtopgm** assumes the pixels in the input go top to bottom, left to right. If you specify **-bt** or **-bottomfirst**, **rawtopgm** assumes the pixels go bottom to top, left to right. The Molecular Dynamics and Leica confocal format, for example, use the latter arrangement.

If you don't specify **-bt** when you should or vice versa, the resulting image is upside down, which you can correct with **pnmflip .**

This option causes **rawtopgm** to read the entire input stream into storage at once, which may take a lot of storage. Ordinarily, **rawtopgm** stores only one row at a time.

For backwards compatibility, **rawtopgm** also accepts **-tb** and **-topbottom** to mean exactly the same thing. The reasons these are named backwards is that the original author thought of it as specifying that the wrong results of assuming the data is top to bottom should be corrected by flipping the result top for bottom. Today, we think of it as simply specifying the format of the input data so that there are no wrong results.

SEE ALSO

pgm(5), **rawtoppm**(1), **pnmflip**(1)

AUTHORS

Copyright (C) 1989 by Jef Poskanzer.
Modified June 1993 by Oliver Trepte, oliver@fysik4.kth.se

NAME

rawtoppm - convert raw RGB bytes into a portable pixmap

SYNOPSIS

rawtoppm [**-headerskip** *N*] [**-rowskip** *N*] [**-rgb**|**-rbg**|**-grb** |**-gbr**|**-brg**|**-bgr**] [**-interpixel**|**-interrow**] *width height* [*imagedata*]

DESCRIPTION

Reads raw RGB bytes as input. Produces a portable pixmap as output. The input file is just RGB bytes. You have to specify the width and height on the command line, since the program obviously can't get them from the file. The maxval is assumed to be 255. If the resulting image is upside down, run it through **pnmflp -tb .**

OPTIONS

-headerskip

If the file has a header, you can use this flag to skip over it.

-rowskip

If there is padding at the ends of the rows, you can skip it with this flag.

-rgb -rbg -grb -gbr -brg -bgr

These flags let you specify alternate color orders. The default is **-rgb**.

-interpixel -interrow

These flags let you specify how the colors are interleaved. The default is **-interpixel**, meaning interleaved by pixel. A byte of red, a byte of green, and a byte of blue, or whatever color order you specified. **-interrow** means interleaved by row - a row of red, a row of green, a row of blue, assuming standard rgb color order. An **-interplane** flag - all the red pixels, then all the green, then all the blue - would be an obvious extension, but is not implemented. You could get the same effect by splitting the file into three parts (perhaps using *dd*), turning each part into a PGM file with rawtopgm, and then combining them with rgb3toppm.

SEE ALSO

ppm(5), rawtopgm(1), rgb3toppm(1), pnmflp(1)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

rgb3toppm - combine three portable graymaps into one portable pixmap

SYNOPSIS

rgb3toppm *redpgmfile greenpgmfile bluepgmfile*

DESCRIPTION

Reads three portable graymaps as input. Combines them and produces one portable pixmap as output.

SEE ALSO

ppmtorgb3(1), pgmtoppm(1), ppmtopgm(1), ppm(5), pgm(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

rgbpalettetoppm − convert ASCII RGB Palette to ppm format

SYNOPSIS

rgbpalettetoppm

DESCRIPTION

Rgbpalettetoppm converts ASCII RGB palettes (each colour described by the numeric values for red, green and blue) to PPM bitmaps. The palette read from standard input and the ppm file is written to standard output.

This program is typically used to colour PGM bitmaps with palettes from Gimp or PlotPlus using the **-map** switch of *pgmtoppm*(1).

AUTHOR

Written 1998 by Michael Haardt (michael@moria.de). This program is put into the public domain.

SEE ALSO

ppm(5), pgmtoppm(1)

NAME

rletopnm − convert a Utah Raster Tools RLE image file into a PNM image file.

SYNOPSIS

rletopnm [**--alphaout=**{*alpha-filename*,**-**}] [**--headerdump**|**-h**] [**--verbose**|**-v**] [**--plain**|**-p**] [*rlefile*|-]

All options may be abbreviated to their minimum unique abbreviation and options and arguments may be in any order.

DESCRIPTION

This program converts Utah Raster Toolkit RLE image files into PNM image files. **rletopnm** handles four types of RLE files: Grayscale (8 bit data, no color map), Pseudocolor (8 bit data with a color map), Truecolor (24 bit data with color map), and Directcolor (24 bit data, no color map). **rletopnm** generates a PPM file for all these cases except for the Grayscale file, for which **rletopnm** generates a PGM file.

rlefile is the RLE input file. If it is absent or **-**, the input comes from Standard Input.

OPTIONS

--alphaout=*alpha-filename*

rletopnm creates a PGM (portable graymap) file containing the alpha channel values in the input image. If the input image doesn't contain an alpha channel, the *alpha-filename* file contains all zero (transparent) alpha values. If you don't specify **--alphaout**, **rletopnm** does not generate an alpha file, and if the input image has an alpha channel, **rletopnm** simply discards it.

If you specify **-** as the filename, **rletopnm** writes the alpha output to Standard Output and discards the image.

See **pnmcomp**(1) for one way to use the alpha output file.

--verbose

This option causes **rletopnm** to operate in verbose mode. It prints messages about what it's doing, including the contents of the RLE image header, to Standard Error.

--headerdump

This option causes **rletopnm** to operate in header dump mode. It prints the contents of the RLE image header to Standard Error, but does not produce any other output.

--plain This option causes the PNM output file to be in the "plain" (text) format, instead of the default "raw" (binary) format. See **ppm**(5) and **pgm**(5) for details on the difference.

EXAMPLES

rletopnm −−verbose lenna.rle >lenna.ppm

While running in verbose mode, convert lenna.rle to PPM format and store the resulting image as lenna.ppm.

rletopnm −−headerdump file.rle

Dump the header information of the RLE file called file.rle.

rletopnm --alphaout=dartalpha.pgm dart.rle >dart.ppm

Convert RLE file dart.rle to PPM format as dart.ppm. Store the alpha channel of dart.rle as dartalpha.pgm (if dart.rle doesn't have an alpha channel, store a fully transparent alpha mask as dartalpha.pgm).

SEE ALSO

pnmtorle(1), **pnmconvol**(1), **pnm**(5), **ppm**(5), **pgm**(5), **urt**(1), **RLE**(5)

AUTHOR

Wes Barris
Army High Performance Computing Research Center (AHPCRC)

Minnesota Supercomputer Center, Inc.

Modifi cations by Eric Haines to support raw and plain formats.

Modifi cations by Bryan Henderson to create alpha fi les and use mnemonic options.

NAME

sbigtopgm − convert an SBIG CCDOPS file into a portable graymap

SYNOPSIS

sbigtopgm [*sbigfile*]

DESCRIPTION

Reads an an image file in the native format used by the Santa Barbara Instrument Group (SBIG) astronomical CCD cameras, and produces a portable graymap as output. Additional information on SBIG cameras and documentation of the file format is available at the Web site:

http://www.sbig.com/

SEE ALSO

pgm(5)

AUTHOR

John Walker (**http://www.fourmilab.ch/**), January 1998. This program is in the public domain.

NAME

sgitopnm - convert a SGI image file to a portable anymap

SYNOPSIS

sgitopnm [**-verbose**] [**-channel** *c*] [*SGIfile*]

DESCRIPTION

Reads an SGI image file as input. Produces a PGM image for a 2-dimensional (1 channel) input file, and a PPM image for a 3-dimensional (3 or more channels) input file.

Alternatively, produces a PGM image of any one of the channels in the input file.

OPTIONS

-verbose
Give some information about the SGI image file.

-channel *c*
Extract channel *c* of the image as a PGM image. Without this option, **sgitopnm** extracts the first 3 channels as a PPM image or, if the input has only 1 channel, extracts that as a PGM image, and if the input has 2 channels, fails.

REFERENCES

SGI Image File Format documentation (draft v0.95) by Paul Haeberli (paul@sgi.com). Available via ftp at sgi.com:graphics/SGIIMAGESPEC.

SEE ALSO

pnm(5), **pnmtosgi**(1)

AUTHOR

Copyright (C) 1994 by Ingo Wilken (Ingo.Wilken@informatik.uni-oldenburg.de)

NAME

sirtopnm - convert a Solitaire file into a portable anymap

SYNOPSIS

sirtopnm [*sirfile*]

DESCRIPTION

Reads a Solitaire Image Recorder file as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's an MGI TYPE 17 file, a *pgm* file is written. If it's an MGI TYPE 11 file, a *ppm* file is written. The program tells you which type it is writing.

BUGS

SEE ALSO

pnmtosir(1), pnm(5)

AUTHOR

Copyright (C) 1991 by Marvin Landis.

NAME

sldtoppm - convert an AutoCAD slide file into a portable pixmap

SYNOPSIS

sldtoppm [**-adjust**] [**-dir**] [**-height**|**-ysize** *s*] [**-info**] [**-lib**|**-Lib** *name*] [**-scale** *s*] [**-verbose**] [**-width**|**-xsize** *s*] [*slidefile*]

DESCRIPTION

Reads an AutoCAD® slide file and outputs a portable pixmap. If no *slidefile* is specified, input is read from standard input. The ppmdraw library is used to convert the vector and polygon information in the slide file to a pixmap; see the file ppmdraw.h for details on this package.

OPTIONS

-adjust If the display on which the slide file was created had non-square pixels, when the slide is processed with **sldtoppm** and the **-adjust** option is not present, the following warning will appear:

Warning - pixels on source screen were non-square.
Specifying **-adjust** will correct image width to compensate.

Specifying the **-adjust** option causes **sldtoppm** to scale the width of the image so that pixels in the resulting portable pixmap are square (and hence circles appear as true circles, not ellipses). The scaling is performed in the vector domain, before scan converting the objects. The results are, therefore, superior in appearance to what you'd obtain were you to perform the equivalent scaling with **pnmscale** after the bitmap had been created.

-dir The input is assumed to be an AutoCAD slide library file. A directory listing each slide in the library is printed on standard error.

-height *size*
Scales the image in the vector domain so it is *size* pixels in height. If no **-width** or **-xsize** option is specified, the width will be adjusted to preserve the pixel aspect ratio.

-info Dump the slide file header on standard error, displaying the original screen size and aspect ratio among other information.

-lib *name*
Extracts the slide with the given *name* from the slide library given as input. The specified *name* is converted to upper case.

-Lib *name*
Extracts the slide with the given *name* from the slide library given as input. The *name* is used exactly as specified; it is not converted to upper case.

-scale *s* Scales the image by factor *s*, which may be any floating point value greater than zero. Scaling is done after aspect ratio adjustment, if any. Since scaling is performed in the vector domain, before rasterisation, the results look much better than running the output of **sldtoppm** through **pnmscale**.

-verbose
Dumps the slide file header and lists every vector and polygon in the file on standard error.

-width *size*
Scales the image in the vector domain so it is *size* pixels wide. If no **-height** or **-ysize** option is specified, the height will be adjusted to preserve the pixel aspect ratio.

-xsize *size*
Scales the image in the vector domain so it is *size* pixels wide. If no **-height** or **-ysize** option is specified, the height will be adjusted to preserve the pixel aspect ratio.

-ysize *size*
Scales the image in the vector domain so it is *size* pixels in height. If no **-width** or **-xsize** option is specified, the width will be adjusted to preserve the pixel aspect ratio.

All flags can be abbreviated to their shortest unique prefix.

BUGS

Only Level 2 slides are converted. Level 1 format has been obsolete since the advent of AutoCAD Release 9 in 1987, and was not portable across machine architectures.

Slide library items with names containing 8 bit (such as ISO) or 16 bit (Kanji, for example) characters may not be found when chosen with the **-lib** option unless **sldtoppm** has been built with character set conversion functions appropriate to the locale. You can always retrieve slides from libraries regardless of the character set by using the **-Lib** option and specifying the precise name of library member. Use the **-dir** option to list the slides in a library if you're unsure of the exact name.

SEE ALSO

AutoCAD Reference Manual: *Slide File Format*, **pnmscale**(1), **ppm**(5)

AUTHOR

John Walker
Autodesk SA
Avenue des Champs-Montants 14b
CH-2074 MARIN
Suisse/Schweiz/Svizzera/Svizra/Switzerland

Usenet:	kelvin@Autodesk.com
Fax:	038/33 88 15
Voice:	038/33 76 33

NAME

spctoppm - convert an Atari compressed Spectrum file into a portable pixmap

SYNOPSIS

spctoppm [*spcfile*]

DESCRIPTION

Reads an Atari compressed Spectrum file as input. Produces a portable pixmap as output.

SEE ALSO

sputoppm(1), ppm(5)

AUTHOR

Copyright (C) 1991 by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

spottopgm − convert SPOT satellite images to Portable Greymap format

SYNTAX

spottopgm [−1|2|3] [Firstcol Firstline Lastcol Lastline] inputfile

OPTIONS

−1|2|3 Extract the given colour from the SPOT image. The colours are infra-red, visible light and ultra-violet, although I don't know which corresponds to which number. If the image is in colour, this will be announced on standard error. The default colour is 1.

Firstcol Firstline Lastcol Lastline
Extract the specified rectangle from the SPOT image. Most SPOT images are 3000 lines long and 3000 or more columns wide. Unfortunately the SPOT format only gives the width and not the length. The width is printed on standard error. The default rectangle is the width of the input image by 3000 lines.

DESCRIPTION

Spottopgm converts the named **inputfile** into Portable Greymap format, defaulting to the first color and the whole SPOT image unless specified by the options.

INSTALLATION

You **must** edit the source program and either define BIG_ENDIAN or LITTLE_ENDIAN, and fix the typedefs for uint32_t, uint16_t and uint8_t appropriately.

BUGS

Currently *spottopgm* doesn't determine the length of the input file; this would involve two passes over the input file. It defaults to 3000 lines instead.

Spottopgm could extract a three-color image (ppm), but I didn't feel like making the program more complicated than it is now. Besides, there is no one-to-one correspondence between red, green, blue and infra-red, visible and ultra-violet.

I've only had a limited number of SPOT images to play with, and therefore wouldn't guarantee that this will work on any other images.

AUTHOR

Warren Toomey wkt@csadfa.cs.adfa.oz.au

SEE ALSO

The rest of the Pbmplus suite.

NAME

sputoppm - convert an Atari uncompressed Spectrum file into a portable pixmap

SYNOPSIS

sputoppm [*spufile*]

DESCRIPTION

Reads an Atari uncompressed Spectrum file as input. Produces a portable pixmap as output.

SEE ALSO

spctoppm(1), ppm(5)

AUTHOR

Copyright (C) 1991 by Steve Belczyk (seb3@gte.com) and Jef Poskanzer.

NAME

tgatoppm − convert TrueVision Targa file into a portable pixmap

SYNOPSIS

tgatoppm [**--alphaout=**{*alpha-filename*,**-**}] [**--headerdump**] *tga-filename*

DESCRIPTION

Reads a TrueVision Targa file as input. Produces a portable pixmap as output.

OPTIONS

--alphaout=*alpha-filename*

tgatoppm creates a PGM (portable graymap) file containing the alpha channel values in the input image. If the input image doesn't contain an alpha channel, the *alpha-filename* file contains all zero (transparent) alpha values. If you don't specify **--alphaout**, **tgatoppm** does not generate an alpha file, and if the input image has an alpha channel, **tgatoppm** simply discards it.

If you specify **-** as the filename, **tgatoppm** writes the alpha output to Standard Output and discards the image.

See **pnmcomp**(1) for one way to use the alpha output file.

--headerdump

Causes the header information to be dumped to stderr.

All options can be abbreviated to their shortest unique prefix. Should really be in PNM, not PPM.

SEE ALSO

ppmtotga(1), **pnmcomp**(1), **ppm**(5)

AUTHOR

Partially based on tga2rast, version 1.0, by Ian J. MacPhedran.

Copyright (C) 1989 by Jef Poskanzer.

NAME

thinkjettopbm − convert HP ThinkJet printer commands file to PBM

SYNOPSIS

thinkjettopbm [-*d*] [*thinkjet_file*]

DESCRIPTION

Reads HP ThinkJet printer commands from the standard input, or *thinkjet_file* if specified, and writes a PBM image to the standard output. Text and non-graphics command sequences are silently ignored.

The **-d** option turns on debugging messages which are written to the standard error stream.

BUGS

Handles only a small subset of ThinkJet command sequences, but enough to convert screen images from older HP test equipment.

SEE ALSO

pbm(5), **pjtoppm**(1)

AUTHOR

Copyright (C) 2001 by W. Eric Norum

NAME

tifftopnm − convert a TIFF file into a portable anymap

SYNOPSIS

tifftopnm [**-alphaout=**{*alpha-filename*,**-**}] [**-headerdump**] [**-respectfillorder**] [*tiff-filename*]

You may abbreviate any option to its shortest unique prefix. You may use two hyphens instead of one in options. You may separate an option and its value either by an equals sign or white space.

DESCRIPTION

Reads a TIFF file as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's black & white, generates a *pbm* file; if it's grayscale, generates a *pgm* file; otherwise, a *ppm* file. The program tells you which type it is writing.

This program cannot read every possible TIFF file -- there are myriad variations of the TIFF format. However, it does understand monochrome and gray scale, RGB, RGBA (red/green/blue with alpha channel), CMYK (Cyan-Magenta-Yellow-Black ink color separation), and color palette TIFF files. An RGB file can have either single plane (interleaved) color or multiple plane format. The program reads 1-8 and 16 bit-per-sample input, the latter in either bigendian or littlendian encoding. Tiff directory information may also be either bigendian or littendian.

One reason this program isn't as general as TIFF programs often are is that it does not use the TIFFRGBAImageGet() function of the TIFF library to read TIFF files. Rather, it uses the more primitive TIFFReadScanLine() function and decodes it itself.

There is no fundamental reason that this program could not read other kinds of TIFF files; the existing limitations are mainly because no one has asked for more.

The PNM output has the same maxval as the Tiff input, except that if the Tiff input is colormapped (which implies a maxval of 65535) the PNM output has a maxval of 255. Though this may result in lost information, such input images hardly ever actually have more color resolution than a maxval of 255 provides and people often cannot deal with PNM files that have maxval > 255. By contrast, a non-colormapped Tiff image that doesn't need a maxval > 255 doesn't *have* a maxval > 255, so when we see a non-colormapped maxval > 255, we take it seriously and produce a matching output maxval.

The *tiff-filename* argument names the regular file that contains the Tiff image. If you specify "-" or don't specify this argument, **tfftopnm** uses Standard Input. In either case, the file must be seekable. That means no pipe, but any regular file is fine.

OPTIONS

-alphaout=*alpha-filename*

tifftopnm creates a PGM (portable graymap) file containing the alpha channel values in the input image. If the input image doesn't contain an alpha channel, the *alpha-filename* file contains all zero (transparent) alpha values. If you don't specify **-alphaout**, **tifftopnm** does not generate an alpha file, and if the input image has an alpha channel, **tifftopnm** simply discards it.

If you specify **-** as the filename, **tifftopnm** writes the alpha output to Standard Output and discards the image.

See **pnmcomp**(1) for one way to use the alpha output file.

-respectfillorder

By default, **tifftopnm** ignores the "fillorder" tag in the TIFF input, which means it may incorrectly interpret the image. To make it follow the spec, use this option. For a lengthy but engaging discussion of why **tifftopnm** works this way and how to use the **-respectfillorder** option, see the note on fillorder below.

-headerdump
Dump TIFF file information to stderr. This information may be useful in debugging TIFF file conversion problems.

All options can be abbreviated to their shortest unique prefix.

NOTES

Fillorder

There is a piece of information in the header of a TIFF image called "fillorder." The TIFF specification quite clearly states that this value tells the order in which bits are arranged in a byte in the description of the image's pixels. There are two options, assuming that the image has a format where more than one pixel can be represented by a single byte: 1) the byte is filled from most signficant bit to least signficant bit going left to right in the image; and 2) the opposite.

However, there is confusion in the world as to the meaning of fillorder. Evidence shows that some people believe it has to do with byte order when a single value is represented by two bytes.

These people cause TIFF images to be created that, while they use a MSB-to-LSB fillorder, have a fillorder tag that says they used LSB-to-MSB. A program that properly interprets a TIFF image will not end up with the image that the author intended in this case.

For a long time, **tifftopnm** did not understand fillorder itself and assumed the fillorder was MSB-to-LSB regardless of the fillorder tag in the TIFF header. And as far as I know, there is no legitimate reason to use a fillorder other than MSB-to-LSB. So users of **tifftopnm** were happily using those TIFF images that had incorrect fillorder tags.

So that those users can continue to be happy, **tifftopnm** today continues to ignore the fillorder tag unless you tell it not to. (It does, however, warn you when the fillorder tag does not say MSB-to-LSB that the tag is being ignored).

If for some reason you have a TIFF image that actually has LSB-to-MSB fillorder, and its fillorder tag correctly indicates that, you must use the **-respectfillorder** option on **tifftopnm** to get proper results.

Examples of incorrect TIFF images are at ftp://weather.noaa.gov. They are apparently created by a program called **faxtotiff**.

This note was written on January 1, 2002.

SEE ALSO

pnmtotiff(1), **pnmtotiffcmyk**(1), **pnmcomp**(1), **pnm**(5)

AUTHOR

Derived by Jef Poskanzer from tif2ras.c, which is Copyright (c) 1990 by Sun Microsystems, Inc. Author: Patrick J. Naughton (naughton@wind.sun.com).

NAME

wbmptopbm - convert a wireless bitmap (wbmp) file to a portable bitmap (pbm)

SYNOPSIS

wbmptopbm [*wbmpfile*]

DESCRIPTION

Reads a wbmp file as input. Produces a portable bitmap as output.

LIMITATIONS

Currently only WBMP type 0 is recognized. This is the only type specified in the WAP 1.1 specifications.

SEE ALSO

pbm(5), **pbmtowbmp**(1), **Wireless Application Environment Specification.**

AUTHOR

Copyright (C) 1999 Terje Sannum <terje@looplab.com>.

NAME

winicontoppm − convert a Windows .ico file into 1 or more portable pixmap files

SYNOPSIS

winicontoppm [−*writeands*] [−*allicons*|−*bestqual*] [−*multippm*] [−*verbose*] [*iconfile*] [*ppmdestfile*]

DESCRIPTION

Reads a Microsoft Windows .ico file, converts it to one or more ppms.

A Windows icon contains 1 or more images, at different resolutions and color depths. Each image has an ’and’ mask, which contains transparancy data.

By default, the output goes to Standard Output. If you specify *ppmdestfile*, output goes into one or more files named as follows. If it’s just one file (i.e. you specify the **-multippm** option or don’t specify **-allicons**), the file specification is *ppmdestfile***.ppm**. If it’s multiple files, their file specifications are *ppmdestfile***_1.ppm**, *ppmdestfile***_2.ppm**, etc.

When you specify the **-writeands** option, the filenames above are modified to include the string **xor** as in *ppmdestfile***_xor.ppm** or *ppmdestfile***_xor_1.ppm**.

OPTIONS

−writeands
: For each icon written, also write the ’and’ (transparancy) mask as a seperate PBM file. It’s name is of the form *ppmdestfile***_and.pbm** or *ppmdestfile***_and_1.pbm**.

−allicons
: Extract all images from the .ico file.

−bestqual
: Extract only the best quality (largest, then highest bpp) image from the .ico file.

−multippm
: Write all ppms to a single file.

SEE ALSO

ppmtowinicon(1), **ppm**(5)

AUTHOR

Copyright (C) 2000 by Lee Benfield.

NAME

xbmtopbm - convert an X11 or X10 bitmap into a portable bitmap

SYNOPSIS

xbmtopbm [*bitmapfile*]

DESCRIPTION

Reads an X11 or X10 bitmap as input. Produces a portable bitmap as output.

SEE ALSO

pbmtoxbm(1), pbmtox10bm(1), pbm(5)

AUTHOR

Copyright (C) 1988 by Jef Poskanzer.

NAME

ximtoppm − convert an Xim file into a portable pixmap

SYNOPSIS

ximtoppm [**--alphaout=**{*alpha-filename*,**-**}] [*ximfile*]

DESCRIPTION

Reads an Xim file as input. Produces a portable pixmap as output. The Xim toolkit is included in the contrib tree of the X.V11R4 release.

OPTIONS

--alphaout=*alpha-filename*

> **ximtoppm** creates a PGM (portable graymap) file containing the alpha channel values in the input image. If the input image doesn't contain an alpha channel, the *alpha-filename* file contains all zero (transparent) alpha values. If you don't specify **--alphaout**, **ximtoppm** does not generate an alpha file, and if the input image has an alpha channel, **ximtoppm** simply discards it.
>
> If you specify **-** as the filename, **ximtoppm** writes the alpha output to Standard Output and discards the image.
>
> Actually, an Xim image can contain an arbitrary fourth channel -- it need not be an Alpha channel. **ximtoppm** extracts any fourth channel it finds as described above; it doesn't matter if it is an alpha channel or not.
>
> See **pnmcomp**(1) for one way to use the alpha output file.

All options can be abbreviated to their shortest unique prefix.

SEE ALSO

pnmcomp(1), **ppm**(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

NAME

xpmtoppm - convert an X11 pixmap into a PPM image

SYNOPSIS

xpmtoppm [**--alphaout=**{*alpha-filename*,**-**}] [**-verbose**] [*xpmfile*]

DESCRIPTION

Reads an X11 pixmap (XPM version 1 or 3) as input. Produces a PPM file as output.

OPTIONS

--alphaout=*alpha-filename*

xpmtoppm creates a PBM file containing the transparency mask for the image. If the input image doesn't contain transparency information, the *alpha-filename* file contains all white (opaque) alpha values. If you don't specify **--alphaout**, **xpmtoppm** does not generate an alpha file, and if the input image has transparency information, **xpmtoppm** simply discards it.

If you specify **-** as the filename, **xpmtoppm** writes the alpha output to Standard Output and discards the image.

See **pnmcomp**(1) for one way to use the alpha output file.

--verbose

xpmtoppm prints information about its processing on Standard Error.

LIMITATIONS

The support to XPM version 3 is limited. Comments can only be single lines and there must be for every pixel a default colorname for a color type visual.

SEE ALSO

ppmtoxpm(1), **pnmcomp**(1), **ppm**(5)

AUTHOR

Copyright (C) 1991 by Jef Poskanzer.

Upgraded to support XPM version 3 by
Arnaud Le Hors (lehors@mirsa.inria.fr)
Tue Apr 9 1991

NAME

xvminitoppm - convert a XV "thumbnail" picture to PPM

SYNOPSIS

xvminitoppm [*xvminipic*]

DESCRIPTION

Reads a XV "thumbnail" picture (a miniature picture generated by the "VisualSchnauzer" browser) as input. Produces a portable pixmap as output.

SEE ALSO

ppm(5), xv(1)

AUTHOR

Copyright (C) 1993 by Ingo Wilken

NAME

xwdtopnm - convert a X11 or X10 window dump file into a portable anymap

SYNOPSIS

xwdtopnm [*xwdfile*]

DESCRIPTION

Reads a X11 or X10 window dump file as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's black & white, a *pbm* file is written, else if it's grayscale a *pgm* file, else a *ppm* file. The program tells you which type it is writing.

Using this program, you can convert anything on an X workstation's screen into an anymap. Just display whatever you're interested in, do an xwd, run it through xwdtopnm, and then use pnmcut to select the part you want.

BUGS

I haven't tested this tool with very many configurations, so there are probably bugs. Please let me know if you find any.

SEE ALSO

pnmtoxwd(1), pnm(5), xwd(1)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

ybmtopbm - convert a Bennet Yee "face" file into a portable bitmap

SYNOPSIS

ybmtopbm [*facefile*]

DESCRIPTION

Reads a file acceptable to the *face* and *xbm* programs by Bennet Yee (bsy+@cs.cmu.edu). Writes a portable bitmap as output.

SEE ALSO

pbmtoybm(1), pbm(5), face(1), face(5), xbm(1)

AUTHOR

Copyright (C) 1991 by Jamie Zawinski and Jef Poskanzer.

NAME

yuvplittoppm - convert a Y- and U- and a V-file into a portable pixmap.

SYNOPSIS

yuvsplittoppm *basename width height* [-ccir601]

DESCRIPTION

Reads three files, containing the YUV components, as input. These files are *basename*.Y, *basename*.U, and *basename*.V. Produces a portable pixmap on stdout.

Since the YUV files are raw files, the dimensions *width* and *height* must be specified on the command line.

OPTIONS

-ccir601

Assumes that the YUV triplets are scaled into the smaller range of the CCIR 601 (MPEG) standard. Else, the JFIF (JPEG) standard is assumed.

SEE ALSO

ppmtoyuvsplit(1), yuvtoppm(1), ppm(5)

AUTHOR

Marcel Wijkstra <wijkstra@fwi.uva.nl>, based on *ppmtoyuvsplit.*

NAME

yuvtoppm - convert Abekas YUV bytes into a portable pixmap

SYNOPSIS

yuvtoppm *width height* [*imagedata*]

DESCRIPTION

Reads raw Abekas YUV bytes as input. Produces a portable pixmap as output. The input file is just YUV bytes. You have to specify the width and height on the command line, since the program obviously can't get them from the file. The maxval is assumed to be 255.

SEE ALSO

ppmtoyuv(1), ppm(5)

AUTHOR

Marc Boucher <marc@PostImage.COM>, based on Example Conversion Program, A60/A64 Digital Video Interface Manual, page 69.

Copyright (C) 1991 by DHD PostImage Inc.

Copyright (C) 1987 by Abekas Video Systems Inc.

NAME

zeisstopnm - convert a Zeiss confocal file into a portable anymap

SYNOPSIS

zeisstopnm [-*pgm* | -*ppm*] [*zeissfile*]

DESCRIPTION

Reads a Zeiss confocal file as input. Produces a portable anymap as output. The type of the output file depends on the input file - if it's grayscale a *pgm* file, else a *ppm* file will be produced. The program tells you which type it is writing.

OPTIONS

-pgm Force the output to be a *pgm* file.

-ppm Force the output to be a *ppm* file.

SEE ALSO

pnm(5)

AUTHOR

Copyright (C) 1993 by Oliver Trepte

NAME

libpbm - functions to read and write PBM image files

SYNOPSIS

#include <pbm.h>

int pm_keymatch(char * *str*, **char *** *keyword*, **int** *minchars*);

int pm_maxvaltobits(int *maxval*);

int pm_bitstomaxval(int *bits*);

unsigned int pm_lcm(unsigned int *x*, **unsigned int** *y*, **unsigned int** *z*, **unsigned int** *limit*);

void pm_message(char * *fmt*, **...);**

void pm_error(char * *fmt*, **...);**

void pm_perror(char * *fmt*, **...);**

void pm_usage(char * *usage*);

FILE *pm_openr(char * *name*)

FILE *pm_openw(char * *name*);

FILE *pm_openr_seekable(const char * *name*);

FILE *pm_close(FILE * *fp*);

char *pm_read_unknown_size(FILE * *fp*, **long ****nread*);

unsigned int pm_tell(FILE * *fileP*);

void pm_seek(FILE * *fileP*, **unsigned long** *filepos*);

bit **pbm_allocarray(int *cols*, **int** *rows*);

bit *pbm_allocrow(int *cols*);

pbm_freearray(bit *****bits*, **int *rows*);

**pbm_freerow(bit ****bitrow*);

void pbm_readpbminit(FILE * *fp*, **int ****colsP*, **int ****rowsP*, **int *** *formatP*);

void pbm_readpbmrow(FILE * *fp*, **bit ****bitrow*, **int** *cols*, **int** *format*);

void pbm_readpbmrow_packed(FILE * *fp*,
unsigned char * const *packed_bits*, **const int** *cols*, **const int** *format*);

void bit pbm_readpbm(FILE *** *fp*, **int ****colsP*, **int ****rowsP*);

void pbm_writepbminit(FILE * *fp*, **int** *cols*, **int** *rows*, **int** *forceplain*);

void pbm_writepbmrow(FILE * *fp*, **bit ****bitrow*, **int** *cols*, **int** *forceplain*);

void pbm_writepbmrow_packed(FILE * *fp*,
unsigned char * const *packed_bits*, **const int** *cols*, **const int** *forceplain*);

void pbm_writepbm(FILE * *fp*, **bit ****bits*, **int** *cols*, **int** *rows*, **int** *forceplain***);**

#define pbm_packed_bytes(*cols***) ...**

void pbm_nextimage(FILE * *file*, **int * const** *eofP***);**

void pbm_check(FILE * *file*, **const enum pm_check_type** *check_type*, **const int** *format*, **const int** *cols*, **const int** *rows*, **enum pm_check_code * const** *retval***);**

int pm_readbigshort(FILE ***in*, **short ***sP*);**

int pm_writebigshort(FILE ***out*, **short *s***);**

int pm_readbiglong(FILE ***in*, **long ***lP*);**

int pm_writebiglong(FILE ***out*, **long *l***);**

int pm_readlittleshort(FILE ***in*, **short ***sP*);**

int pm_writelittleshort(FILE ***out*, **short *s***);**

int pm_readlittlelong(FILE ***in*, **long ***lP*);**

int pm_writelittlelong(FILE ***out*, **long *l***);**

DESCRIPTION - PACKAGE-WIDE ROUTINES

KEYWORD MATCHING

pm_keymatch() does a case-insensitive match of **str** against **keyword**. **str** can be a leading sunstring of **keyword**, but at least **minchars** must be present.

MAXVAL ARITHMETIC

pm_maxvaltobits() and **pm_bitstomaxval()** convert between a maxval and the minimum number of bits required to hold it.

pm_lcm() computes the least common multiple of 3 integers. You also specify a limit and if the LCM would be higher than that limit, **pm_lcm()** just returns that limit.

MESSAGES AND ERRORS

pm_message() is a **printf()** style routine to write an informational message to the Standard Error file stream. **pm_message()** suppresses the message, however, if the user specified the **-quiet** option on the command line. See the initialization functions, e.g. **pbm_init()**, for information on the **-quiet** option. Note that Netpbm programs are often used interactively, but also often used by programs. In the interactive case, it is nice to issue messages about what the program is doing, but in the program case, such messages are usually undesirable. By using **pm_message()** for all your messages, you make your program usable in both cases. Without any effort on your part, program users of your program can avoid the messages by specifying the **-quiet** option.

pm_error() is a **printf()** style routine that writes an error message to the Standard Error file stream and exits the program with an exit code of 1.

GENERIC FILE ACCESS

pm_openr() opens the given file for reading, with appropriate error checking. A filename of **-** is taken to mean Standard Input. **pm_openw()** opens the given file for writing, with appropriate error checking. **pm_close()** closes the file descriptor, with appropriate error checking.

pm_openr_seekable() appears to open the file just like **pm_openr()**, but the file thus opened is guaranteed to be seekable (you can use ftell() and fseek() on it). **pm_openr_seekable()** pulls this off by copying the entire file to a temporary file and giving you the handle of the temporary file, if it has to. If the file you name is a regular file, it's already seekable so **pm_openr_seekable()** just does the same thing as **pm_openr()**. But if it is, say, a pipe, it isn't seekable. So **pm_openr_seekable()** reads the pipe until EOF into a temporary file, then opens that temporary file and returns the handle of the temporary file. The temporary file is seekable.

The file **pm_openr_seekable()** creates is one that the operating system recognizes as temporary, so when you close the file, by any means, it gets deleted.

You need a seekable file if you intend to make multiple passes through the file. The only alternative is to read the entire image into memory and work from that copy. That may use too much memory. Note that the image takes less space in the file cache than in a buffer in memory. As much as 96 times less space! Each sample is an integer in the buffer, which is usually 96 bits. In the file, a sample may be as small as 1 bit and rarely more than 8 bits.

pm_read_unknown_size() reads an entire file or input stream of unknown size to a buffer. Allocate memory more memory as needed. The calling routine has to free the allocated buffer with **free()**.

pm_read_unknown_size() returns a pointer to the allocated buffer. The **nread** argument returns the number of bytes read.

pm_tell() returns a handle for the current position of the file, whether it be the header or a row of the raster. Use the handle as an argument to **pm_seek()** to reposition the file there later. The file must be seekable (which you can ensure by opening it with **pm_openr_seekable()**)**or**this**may**fail.

ENDIAN I/O

pm_readbigshort(), **pm_writebigshort()**, **pm_readbiglong()**, **pm_writebiglong()**, **pm_readlittleshort()**, **pm_writelittleshort()**, **pm_readlittlelong()**, and **pm_writelittlelong()** are routines to read and write short and long ints in either big- or little-endian byte order. The return value is **0** upon success and **-1** upon failure (either EOF or I/O error).

DESCRIPTION - PBM-SPECIFIC ROUTINES

TYPES AND CONSTANTS

typedef ... bit;

#define PBM_WHITE ...

#define PBM_BLACK ...

Each **bit** should contain only the values of **PBM_WHITE** or **PBM_BLACK**.

#define PBM_FORMAT ...

#define RPBM_FORMAT ...

#define PBM_TYPE PBM_FORMAT

#define PBM_FORMAT_TYPE(*f* **) ...**

These are for distinguishing different file formats and types.

INITIALIZATION

All PBM programs must call **pbm_init** just after invocation, before processing arguments.

MEMORY MANAGEMENT

pbm_allocarray() allocates an array of bits. **pbm_allocrow()** allocates a row of the given number of bits. **pbm_freearray()** frees the array allocated with **pbm_allocarray()** containing the given number of rows. **pbm_freerow()** frees a row of bits.

READING PBM IMAGE FILES

pbm_readpbminit() reads the header from a PBM image in a PBM file, filling in the rows, cols and format variables. **pbm_readpbmrow()** reads a row of bits into the *bitrow* array. Format and cols were filled in by **pbm_readpbminit()**. **pbm_readpbmrow_packed()** is like **pbm_readrow()** except instead of returning a **bits** array, it returns an array *packed_bits* of bytes with the pixels of the image row packed into them. The pixels are in order from left to right across the row and from the beginning of the array to the end. Within a byte, the bits are in order from the most significant bit to the least significant bit. If the number of pixels in the row is not a multiple of 8, the last byte returned is padded on the least signficant bit side with undefined bits. White is represented by a **PBM_WHITE** bit; black by **PBM_BLACK**.

pbm_readpbm() reads an entire bitmap file into memory, returning the allocated array and filling in the rows and cols variables. This function combines **pbm_readpbminit()**, **pbm_allocarray()** and **pbm_readpbmrow()**.

WRITING PBM IMAGE FILES

pbm_writepbminit() writes the header for a PBM image in a PBM file. *forceplain* is a boolean value specifying that a plain format (text) file to be written, as opposed to a raw format (binary) one. **pbm_writepbmrow()** writes a row to a PBM file. **pbm_writepbmrow_packed()** is the same as **pbm_writepbmrow()** except that you supply the row to write as an array of bytes packed with bits instead of as a **bits** array. The format of *packed_bits* is the same as that returned by **pbm_readpbmrow()**.

pbm_writepbm() writes the header and all data for a PBM image to a PBM file. This function combines **pbm_writepbminit()** and **pbm_writepbmrow()**.

MISCELLANEOUS

pbm_nextimage() positions a PBM input file to the next image in it (so that a subsequent **pbm_readpbminit()** reads its header).

Immediately before a call to **pbm_nextimage()**, the file must be positioned either at its beginning (i.e. nothing has been read from the file yet) or just after an image (i.e. as left by a **pbm_readpbmrow()** of the last row in the image).

In effect, then, all **pbm_nextimage()** does is test whether there is a next image or the file is positioned at end-of-file.

If **pbm_nextimage()** successfully positions to the next image, it returns ***eofP** false (0). If there is no next image in the file, it returns ***eofP** true (1). If it can't position or determine the file status due to a file error, it issues an error message and exits the program with an error exit code.

pbm_check() checks for the common file integrity error where the file is the wrong size to contain all the image data. **pbm_check()** assumes the file is positioned after an image header (as if **pbm_readpbminit()** was the last operation on the file). It checks the file size to see if the number of bytes left in the file are the number required to contain the image raster. If the file is too short, **pbm_check()** causes the program to exit with an error message and error completion code. Otherwise, it returns one of the following values (enumerations of the **enum pm_check_code** type) as ***retval**:

PM_CHECK_OK
The file's size is exactly what is required to hold the image raster.

PM_CHECK_UNKNOWN_TYPE
format is not a format whose size **pbm_check()** can anticipate. The only format with which **pbm_check()** can deal is raw PBM format.

PM_CHECK_TOO_LONG
The file is longer than it needs to be to contain the image raster. The extra data might be another image.

PM_CHECK_UNCHECKABLE
The file is not a kind that has a predictable size, so there is no simple way for **pbm_check()** to know if it is the right size. Only a regular file has predictable size. A pipe is a common example of a file that does not.

check_type must have the value **PM_CHECK_BASIC** (an enumerated value of the **pm_check_type** enumerated type). Otherwise, the effect of **pbm_check()** is unpredictable. This argument exists for future backward compatible expansion of the function of **pbm_check()**.

SEE ALSO

libpgm(3), **libppm**(3), **libpnm**(3), **pbm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Tony Hansen and Jef Poskanzer.

NAME

libpgm - functions to support portable graymap (PGM) programs

SYNOPSIS

#include <pgm.h>

void pgm_init(int ****argcP*, char ****argv***[]);**

gray ** pgm_allocarray(int *cols***, int** *rows* **);**

gray * pgm_allocrow(int*cols* **);**

void pgm_freearray(gray *****grays*, int***rows* **);**

void pgm_freerow(gray ****grayrow*);**

void pgm_readpgminit(FILE * *fp***, int ****colsP***, int ****rowsP***, gray ****maxvalP***, int *** *formatP* **);**

void pgm_readpgmrow(FILE * *fp***, gray ****grayrow***, int** *cols***, gray** *maxval***, int** *format* **);**

gray ** pgm_readpgm(FILE * *fp***, int ****colsP***, int ****rowsP***, gray ****maxvalP* **);**

void pgm_writepgminit(FILE * fp , int *cols***, int** *rows***, gray** *maxval***, int** *forceplain* **);**

void pgm_writepgmrow(FILE * *fp***, gray ****grayrow***, int** *cols***, gray** *maxval***, int** *forceplain* **);**

void pgm_writepgm(FILE * *fp***, gray **** *grays***, int** *cols***, int** *rows***, gray** *maxval***, int** *forceplain* **);**

void pgm_writepgm(FILE * *fp***, gray *****grays***, int** *cols***, int** *rows***, gray** *maxval***, int** *forceplain* **);**

void pgm_nextimage(FILE * *fi le***, int * const** *eofP***);**

void pgm_check(FILE * *fi le***, const enum pm_check_type** *check_type***, const int** *format***, const int** *cols***, const int** *rows***, const int** *maxval***, enum pm_check_code * const** *retval***);**

typedef ... gray;

#defi ne PGM_MAXMAXVAL ...

#defi ne PGM_OVERALLMAXVAL ...

extern gray pgm_pbmmaxval;

#defi ne PGM_FORMAT ...

#defi ne RPGM_FORMAT ...

#defi ne PGM_TYPE PGM_FORMAT

#defi ne PGM_FORMAT_TYPE(*format***) ...**

DESCRIPTION

TYPES AND CONSTANTS

Each **gray** should contain only the values between **0** and **PGM_OVERALLMAXVAL**. **pgm_pbm-maxval** is the maxval used when a PGM program reads a PBM fi le. Normally it is 1; however, for some programs, a larger value gives better results.

PGM_OVERALLMAXVAL is the maximum value of a maxval in a PGM file. **PGM_MAXMAXVAL** is the maximum value of a maxval in a PGM file that is compatible with the PGM format as it existed before April 2000. It is also the maximum value of a maxval that results in the minimum possible raster size for a particular image. I.e an image with a maxval higher than **PGM_MAXMAXVAL** cannot be read or generated by old PGM processing programs and requires more file space.

PGM_FORMAT is the format code for a Plain PGM format image file. **RPGM_FORMAT** is the format code for a Raw PGM format image file. **PGM_TYPE** is the format type code for the PGM formats. **PGM_FORMAT_TYPE** is a macro that generates code to compute the format type code of a PBM or PGM format from the format code which is its argument.

INITIALIZATION

All PGM programs must call **pgm_init()** just after invocation, before they process their arguments.

MEMORY MANAGEMENT

pgm_allocarray() allocates an array of grays.

pgm_allocrow() allocates a row of the given number of grays.

pgm_freearray() frees the array allocated with **pgm_allocarray()** containing the given number of rows.

pgm_freerow() frees a row of grays allocated with **pgm_allocrow()**.

READING FILES

If a function in this section is called on a PBM format file, it translates the PBM file into a PGM file on the fly and functions as if it were called on the equivalent PGM file. The *format* value returned by **pgm_readpgminit()** is, however, not translated. It represents the actual format of the PBM file.

pgm_readpgminit() reads the header of a PGM file, returning all the information from the header and leaving the file positioned just after the header.

pgm_readpgmrow() reads a row of grays into the *grayrow* array. *format*, *cols*, and *maxval* are the values returned by **pgm_readpgminit()**.

pgm_readpgm() reads an entire PGM image into memory, returning the allocated array as its return value and returning the information from the header as *rows*, *cols*, and *maxval*. This function combines **pgm_readpgminit()**, **pgm_allocarray()**, and **pgm_readpgmrow()**.

WRITING FILES

pgm_writepgminit() writes the header for a PGM file and leaves it positioned just after the header.

forceplain is a logical value that tells **pgm_writepgminit()** to write a header for a plain PGM format file, as opposed to a raw PGM format file.

pgm_writepgmrow() writes the row *grayrow* to a PGM file. For meaningful results, *cols*, *maxval*, and *forceplain* must be the same as was used with **pgm_writepgminit()**.

pgm_writepgm() write the header and all data for a PGM image. This function combines **pgm_writepgminit()** and **pgm_writepgmrow()**.

MISCELLANEOUS

pgm_nextimage() positions a PGM input file to the next image in it (so that a subsequent **pgm_readpgminit()** reads its header).

pgm_nextimage() is analogous to **pbm_nextimage()**, but works on PGM and PBM files.

pgm_check() checks for the common file integrity error where the file is the wrong size to contain all the image data.

pgm_check() is analogous to **pbm_check()**, but works on PGM and PBM files.

SEE ALSO

libpbm(3), **libppm**(3), **libpnm**(3)

NAME

libpnm - functions to support pnm and pam programs

EXAMPLE

```
/* Example program fragment to read a PAM or PNM image
   from stdin, add up the values of every sample in it
   (I don’t know why), and write the image unchanged to
   stdout.  */

#include <pam.h>

struct pam inpam, outpam; unsigned int row;

pnm_init(&argc, argv);

pnm_readpaminit(stdin, &inpam, sizeof(inpam));

outpam = inpam; outpam.fi le = stdout;

pnm_writepaminit(&outpam);

tuplerow = pnm_allocpamrow(&inpam);

for (row = 0; row < inpam.height; row++) {
    unsigned int col;
    pnm_readpamrow(&inpam, tuplerow);
    for (column = 0; column < inpam.width; column++) {
        unsigned int plane;
        for (plane = 0; plane < inpam.depth; column++) {
            grand_total += tuplerow[row][column][plane];
        }
    }
    pnm_writepamrow(&outpam, tuplerow); }

pnm_freepamrow(tuplerow);
```

SYNOPSIS

#include <pnm.h>

void pnm_init(int ****argcP*, char ****argv***[]);**

tuple ** pnm_allocpamarray(struct pam ****pamP*);**

xel ** pnm_allocarray(int *cols***, int** *rows***);**

tuple * pnm_allocpamrow(struct pam ****pamP*);**

xel * pnm_allocrow(int *cols***);**

void pnm_freepamarray(tuple *****tuplearray*, struct pam ****pamP***);**

void pnm_freearray(xel *****xels*, int** *rows***);**

void pnm_freepamrow(tuple ****tuplerow*);**

void pnm_freerow(xel ***_xelrow_);**

tuple * allocpamtuple(struct pam ***_pamP_);**

void pnm_freepamtuple(tuple _tuple_ **);**

void pnm_readpaminit(FILE * _fi le_**, struct pam ***_pamP_**, int** _size_**);**

void pnm_readpnminit(FILE * _fp_**, int ***_colsP_**, int ***_rowsP_**, xelval ***_maxvalP_**, int *** _formatP_ **);**

void pnm_readpamrow(struct pam ***_pamP_, tuple ***_tuplerow_**);**

void pnm_readpnmrow(FILE * _fp_**, xel ***_xelrow_**, int** _cols_**,**
xelval _maxval_**, int** _format_ **);**

tuple ** pnm_readpam(FILE * _fi le_**, struct pam ***_pamP_**,**
int _size_**);**

xel ** pnm_readpnm(FILE * _fp_**, int ***_colsP_**, int ***_rowsP_**,**
xelval ***_maxvalP_,** _int_***** _formatP_ **);"**

void pnm_writepaminit(struct pam ***_pamP_);**

void pnm_writepnminit(FILE * fp , int _cols_**, int** _rows_**, xelval** _maxval_**, int** _format_**, int** _forceplain_**);**

void pnm_writepamrow(struct pam ***_pamP_, const tuple ***_tuplerow_**);**

void pnm_writepnmrow(FILE * _fp_**, xel ***_xelrow_**, int** _cols_**, xelval** _maxval_**, int** _format_**, int** _forceplain_
);

void pnm_writepam(struct pam ***_pamP_, const tuple * const ***_tuplearray_**);**

void pnm_writepnm(FILE * _fp_**, xel **** _xels_**, int** _cols_**, int** _rows_**, xelval** _maxval_**, int** _format_**, int** _force-plain_ **);**

void pnm_checkpam(struct pam ***_pamP_, const enum pm_check_type** _check_type_**, enum pm_check_code ***_retvalP_**);**

void pnm_nextimage(FILE * _fi le_**, int * const** _eofP_**);**

void pnm_check(FILE * _fi le_**, const enum pm_check_type** _check_type_**, const int** _format_**, const int** _cols_**, const int** _rows_**, const xelval** _maxval_**, enum pm_check_code ***_retvalP_**);**

void pnm_promoteformatrow(xel ***_xelrow_, int** _cols_**, xelval** _maxval_**, int** _format_**, xelval** _newmaxval_**, int** _newformat_**);**

void pnm_promoteformatrow(xel ****_xels_, int** _cols_**, xelval** _maxval_**, int** _format_**, xelval** _newmaxval_**, int** _newformat_**);**

xel pnm_whitexel(xelval _maxval_**, int** _format_**);**

xel pnm_blackxel(xelval _maxval_**, int** _format_**);**

void pnm_invertxel(xel ***_x_, xelval** _maxval_**, int** _format_**);**

xel pnm_backgroundxelrow(xel ***_xelrow_, int** _cols_**, xelval** _maxval_**, int** _format_**);**

xel pnm_backgroundxel(xel ****_xels_, int** _cols_**, int** _rows_**, xelval** _maxval_**, int** _format_**);**

void pnm_YCbCrtuple(tuple*tuple***, double ****YP***, double ****CrP***, double ****CbP***);**

struct pam {
int *size*
int *len*
FILE * *fi le*
int *format*
int *plainformat*
int *height*
int *width*
int *depth*
sample *maxval*
int *bytes_per_sample*
char *tuple_type***[256]; }**

typedef ... sample;

typedef ... tuple;

typedef ... xelval;

typedef ... xel;

extern xelval pnm_pbmmaxval;

#defi ne PNM_MAXMAXVAL ...

#defi ne PNM_OVERALLMAXVAL ...

#defi ne PNM_FORMAT ...

#defi ne PNM_ASSIGN1(*x***,***v***) ...**

#defi ne PNM_GET1(*x***) ...**

#defi ne PNM_EQUAL(*x***,***y***) ...**

#defi ne PAM_FORMAT_TYPE(*format***) ...**

#defi ne PNM_FORMAT_TYPE(*format***) ...**

DESCRIPTION

PAM VERSUS PNM FUNCTIONS

The PNM library contains two classes of functions: The pam functions and the pnm functions. The pam functions are enhancements of the pnm functions and you should use them unless you need to be compatible with older PNM libraries that don't have them (those released before August 2000).

The pnm functions operate on PBM, PGM, and PPM images and fi les. They are similar to the functions in the PBM, PGM, and PPM libraries, except the pnm functions let you operate on all three, both reading and writing, without a lot of concern for which of the three formats you are processing.

The pam functions provide all the same functions for operating on PBM, PGM, and PPM libraries, but also operate on the newer PAM images and fi les. The pam functions are easier to use than the pnm functions due to improved parameter lists.

There is no separate PAM library specifi c to the PAM format, as there is for PBM, PGM, and PPM.

THE pam STRUCTURE

The pam functions take most of their arguments in the form of a single **pam** structure. This is not an opaque object, but just a convenient way to organize the information upon which most the functions depend. So you are free to access or set the elements of the structure however you want. But you will find in most cases it is most convenient to call **pnm_readpaminit()** or **pnm_writepaminit()** to set the fields in the **pam** structure before calling any other pam functions, and then just to pass the structure unchanged in all future calls to pam functions.

The fields are:

size The storage size in bytes of this entire structure.

len The length, in bytes, of the information in this structure. The information starts in the first byte and is contiguous. This cannot be greater than **size**. **size** and **len** can be used to make programs compatible with newer and older versions of the Netpbm libraries.

file The file.

format The format code of the raw image. This is **PAM_FORMAT** unless the PAM image is really a view of a PBM, PGM, or PPM image. Then it's **PBM_FORMAT**, **RPBM_FORMAT**, etc.

plainformat
This is a boolean value and means: The format above is a plain (text) format as opposed to a raw (binary) format. This is entirely redundant with the **format** member and exists as a separate member only for computational speed.

height The height of the image in rows.

width The width of the image in number of columns (tuples per row).

depth The depth of the image (degree of or number of samples in each tuple).

maxval
The maxval of the image. See definitions in pam(5).

bytes_per_sample
The number of bytes used to represent each sample in the image file. See the format definition in pam(5). This is entirely redundant with **maxval**. It exists as a separate member for computational speed.

tuple_type
The tuple type of the image. See definitions in pam(5). Netpbm does not define any values for this except the following, which are used for a PAM image which is really a view of a PBM, PGM, or PPM image: **PAM_PBM_TUPLETYPE**, **PAM_PGM_TUPLETYPE**, **PAM_PPM_TUPLETYPE**.

PLAIN VERSUS RAW FORMAT

The PNM formats each come in two varieties: the older plain (text) format and the newer raw (binary) format. There are different format codes for the plain and raw formats, but which of the two formats the pnm and pam functions write is independent of the format code you pass to them.

The pam functions always write raw formats. If you specify the format code for a plain format, a pam function assumes instead the raw version of that format.

The pnm functions choose between plain and raw based on the *forceplain* parameter that every write-type pnm function has. If this boolean value is true, the function writes the plain version of the format specified by the format code. If it is false, the function writes the raw version of the format specified by the format code.

We are trying to stamp out the older plain formats, so it would be a wise choice not to write a program that sets *forceplain* true under any circumstance. A user who needs a plain format can use the **pnmtoplainpnm** program to convert the output of your program to plain format.

PNM TYPES AND CONSTANTS

Each **xel** contains three **xelval**s, each of which should contain only the values between **0** and **PNM_MAXMAXVAL**, inclusive. **pnm_pbmmaxval** is the maxval used when a PNM program reads a PBM file. Normally it is 1; however, for some programs, a larger value gives better results.

PNM XEL MANIPULATIONS

The **PNM_GET1** macro extracts a single value from an xel, when you know it's from a PBM or PGM file. When it's from a PPM file, use **PPM_GETR()**, **PPM_GETG()**, and **PPM_GETB()**.

The **PNM_ASSIGN1** macro assigns a single value to an xel, when you know it's from a PBM or PGM file. When it's from a PPM file, use **PPM_ASSIGN**. The **PNM_EQUAL** macro checks two xels for equality. The **PNM_FORMAT_TYPE** and **PAM_FORMAT_TYPE** macros compute a format type code from a format code. The format types are PBM, PGM, PPM, and PAM. But note that PBM, PGM, and PPM each are two different formats: a plain one and a raw one. So there are four format types, but seven formats. **PNM_FORMAT_TYPE** does not work on the PAM format code.

INITIALIZATION

All PNM and PAM programs must call **pnm_init()** just after startup, before they process their arguments.

pnm_init(), among other things, processes Netpbm universal parameters and removes them from the parameter list.

MEMORY MANAGEMENT

pnm_allocpamarray() allocates space for an array of tuples. **pnm_freepamarray()** frees an array space allocated by **pnm_allocpamarray()** or **pnm_readpam()**.

pnm_allocarray() allocates space for an array of xels. **pnm_freearray()** frees an array space allocated by **pnm_allocarray()** or **pnm_readpnm()**.

pnm_allocpamrow() allocates space for a row of a PAM image. **pnm_freepamrow()** frees it.

pnm_allocrow() allocates space for a row of a PNM image. **pnm_freerow()** frees it.

READING PNM FILES

pnm_readpaminit() reads the header of a PAM or PNM image. It returns the information from the header in the ***pamP** structure. It does not require any members of ***pamP** to be set at invocation, and sets every member. *size* is the storage size of the ***pamP** structure, normally **sizeof(struct pam)**.

The function expects to find the image file positioned to the start of the header and leaves it positioned to the start of the raster.

pnm_readpnminit() is similar to **pnm_readpaminit()**, but reads only PNM images and has a different parameter list.

pnm_readpamrow() reads a row of the raster from a PAM or PNM image file. It expects all of the members of the ***pamP** structure to be set upon invocation and does not modify any of them. It expects to find the file positioned to the start of the row in question in the raster and leaves it positioned just after it. It returns the row as the array of tuples *tuplerow*, which must already have its column pointers set up so that it forms a C 2-dimensional array. The leftmost tuple is Element 0 of this array.

pnm_readpnmrow() is similar to **pnm_readpamrow()** but only works on PNM images and has a different parameter list and returns the row as an array of xels instead of tuples.

pnm_readpam() reads an entire image from a PAM or PNM image file and allocates the space in which to return the raster. It expects to find the file positioned to the first byte of the image and leaves it positioned just after the image.

The function does not require ***pamP** to have any of its members set and sets them all. *size* is the storage size in bytes of the ***pamP** structure, normally **sizeof(struct pam)**.

The return value is a newly allocated array of the rows of the image, with the top row being Element 0 of the array. Each row is represented as **pnm_readpamrow()** would return.

The return value is also effectively a 3-dimensional C array of samples, with the dimensions corresponding to the height, width, and depth of the image, in that order.

pnm_readpam() combines the functions of **pnm_allocpamarray()**, **pnm_readpaminit()**, and iterations of **pnm_readpamrow()**. It may require more dynamic storage than you can afford.

pnm_readpnm() is similar to **pnm_readpam()** except that it reads only PNM images and uses a different parameter list and returns an array of rows such that **pnm_readpnmrow()** would return rather than such that **pnm_readpamrow()** would return.

WRITING FILES

pnm_writepnminit() writes the header of a PAM or PNM image and computes some of the fi elds of the pam structure.

The following members of the ***pamP** structure must be set upon invocation to tell the function how and what to write. **size**, **len**, **fi le**, **format**, **height**, **width**, **depth**, **maxval**, **tuple_type**.

pnm_writepaminit() sets the **plainformat** and **bytes_per_sample** members based on the information supplied.

pnm_writepnminit() is similar to **pnm_writepaminit()** except that it can write only a PNM header and has a different parameter list.

See the description of *forceplain* above.

pnm_writepamrow() writes a row of the raster into a PAM or PNM image fi le. It expects to fi nd the fi le positioned where the row should start and leaves it positioned just after the row. The function requires all the elements of ***pamP** to be set upon invocation and doesn't modify them.

tuplerow is an array of tuples representing the row. The leftmost tuple is Element 0 of this array.

pnm_writepnmrow() is similar to **pnm_writepamrow()** except that it works only on PNM images and has a different parameter list and takes an array of xels instead of an array of tuples. See the description of *forceplain* above.

pnm_writepam() writes an entire PAM or PNM image to a PAM or PNM image fi le. It expects to fi nd the fi le positioned to where the image should start and leaves it positioned just after the image.

The following members of the ***pamP** structure must be set upon invocation to tell the function how and what to write: **size**, **len**, **fi le**, **format**, **height**, **width**, **depth**, **maxval**, **tuple_type**.

pnm_writepam() sets the **plainformat** and **bytes_per_sample** members based on the information supplied.

tuplearray is an array of rows such that you would pass to **pnm_writepamrow()**, with the top row being Element 0 of the array.

pnm_writepam() combines the functions of **pnm_writepaminit()**, and iterations of **pnm_writepamrow()**. It's raster input may be more storage than you can afford.

pnm_writepnm() is similar to **pnm_writepam()** except that it works only on PNM image, has a different parameter list, and takes an array of rows of xels instead of an array of rows of tuples. See the description of *forceplain* above.

MISCELLANEOUS

pnm_nextimage() positions a PNM input file to the next image in it (so that a subsequent **pnm_readpnminit()** reads its header).

pnm_nextimage() is identical to **pbm_nextimage()**.

pam_check() checks for the common file integrity error where the file is the wrong size to contain the raster, according to the information in the header. This works on PAM and PNM images.

pnm_check() is similar to **pam_check()** except it works only on PNM images.

pnm_check() is identical to **ppm_check()**.

PNM FORMAT PROMOTION

pnm_promoteformatrow() promotes a row of xels from one maxval and format to a new set. Use this when you are combining multiple anymaps of different types - just take the maximum of the maxvals and the maximum of the formats, and promote them all to that.

pnm_promoteformat() promotes an entire anymap.

PNM XEL MANIPULATION

pnm_whitexel() and **pnm_blackxel()** return a white or black xel, respectively, for the given *maxval* and *format*.

pnm_invertxel() inverts an xel.

pnm_backgroundxelrow() figures out an appropriate background xel based on the row of xels *xelrow*, which is *cols* xels wide, has maxval *maxval*, and represents an image with format *format*.

This estimate works best when the row is the top or bottom row of the image.

pnm_backgroundxel() does the same thing as **pnm_backgroundxelrow()**, except based on an entire image instead of just one row. This tends to do a slightly better job than **pnmbackgroundxelrow()**.

pnm_YCbCrtuple() Returns the Y/Cb/Cr luminance/chrominance representation of the color represented by the input tuple, assuming that the tuple is an RGB color representation (which is the case if it was read from a PPM image). The output components are based on the same scale (maxval) as the input tuple, but are floating point nonetheless to avoid losing information due to rounding. Divide them by the maxval to get normalized [0..1] values.

SEE ALSO

pbm(5), **pgm**(5), **ppm**(5), **pam**(5), **libpbm**(3), **libpgm**(3), **libppm**(3)

AUTHOR

Copyright (C) 1989, 1991 by Tony Hansen and Jef Poskanzer.

NAME

libppm - functions to support portable pixmap (PPM) programs

SYNOPSIS

#include <ppm.h>

void ppm_init(int ***_argcP_, char ***_argv_**[]);**

pixel ** ppm_allocarray(int _cols_**, int** _rows_ **);**

pixel * ppm_allocrow(int _cols_ **);**

void ppm_freearray(pixel ****_pixels_, int** _rows_ **);**

void ppm_freerow(pixel ***_pixelrow_);**

void ppm_readppminit(FILE * _fp_**, int ***_colsP_**, int ***_rowsP_**, pixval ***_maxvalP_**, int *** _formatP_ **);**

void ppm_readppmrow(FILE * _fp_**, pixel ***_pixelrow_**, int** _cols_**, pixval** _maxval_**, int** _format_ **);**

pixel ** ppm_readppm(FILE * _fp_**, int ***_colsP_**, int ***_rowsP_**, pixvalP ***_maxvalP_ **);**

void ppm_writeppminit(FILE * fp , int _cols_**, int** _rows_**, pixval** _maxval_**, int** _forceplain_ **);**

void ppm_writeppmrow(FILE * _fp_**, pixel ***_pixelrow_**, int** _cols_**, pixval** _maxval_**, int** _forceplain_ **);**

void ppm_writeppm(FILE * _fp_**, pixel **** _pixels_**, int** _cols_**, int** _rows_**, pixval** _maxval_**, int** _forceplain_ **);**

void ppm_writeppm(FILE * _fp_**, pixel ****_pixels_**, int** _cols_**, int** _rows_**, pixval** _maxval_**, int** _forceplain_ **);**

void ppm_nextimage(FILE * _fi le_**, int * const** _eofP_**);**

void ppm_check(FILE * _fi le_**, const enum pm_check_type** _check_type_**, const int** _format_**, const int** _cols_**, const int** _rows_**, const int** _maxval_**,**
enum pm_check_code * const _retval_**);**

typedef ... pixel; typedef ... pixval;

#defi ne PPM_MAXMAXVAL ...

#defi ne PPM_OVERALLMAXVAL ...

#defi ne PPM_FORMAT ...

#defi ne RPPM_FORMAT ...

#defi ne PPM_TYPE PPM_FORMAT

#defi ne PPM_FORMAT_TYPE(_format_**) ...**

extern pixval ppm_pbmmaxval;

pixval PPM_GETR(pixel _p_**) pixval PPM_GETG(pixel** _p_**) pixval PPM_GETB(pixel** _p_**)**

void PPM_ASSIGN(pixel _p_**, pixval** _red_**, pixval** _grn_**, pixval** _blu_**)**

int PPM_EQUAL(pixel _p_**, pixel** _q_**)**

void PPM_DEPTH(pixel *newp***, pixel** *p***, pixval** *oldmaxval***, pixval** *newmaxval***)**

fbat PPM_LUMIN(pixel *p***)**

fbat PPM_CHROM_R(pixel *p***)**

fbat PPM_CHROM_B(pixel *p***)**

pixel ppm_parsecolor(char ****colorname*, pixval** *maxval***)**

char * ppm_colorname(pixel ****colorP*, pixval** *maxval***, int** *hexok***)**

DESCRIPTION

TYPES AND CONSTANTS

Each **pixel** contains three **pixval**s, each of which should contain only the values between **0** and **PPM_MAXMAXVAL**. **ppm_pbmmaxval** is the maxval used when a PPM program reads a PBM file. Normally it is 1; however, for some programs, a larger value gives better results.

MANIPULATING PIXELS

The macros **PPM_GETR**, **PPM_GETG**, and **PPM_GETB** retrieve the red, green, or blue sample, respectively, from the given pixel.

The **PPM_ASSIGN** macro assigns the given values to the red, green, and blue samples of the given pixel.

The **PPM_EQUAL** macro tests two pixels for equality.

The **PPM_DEPTH** macro scales the colors of pixel *p* according the old and new maxvals and assigns the new values to *newp*. It is intended to make writing ppmtowhatever easier.

The **PPM_LUMIN**, **PPM_CHROM_R**, and **PPM_CHROM_B**, macros determine the luminance, red chrominance, and blue chrominance, respectively, of the pixel *p*. The scale of all these values is the same as the scale of the input samples (i.e. 0 to maxval for luminance, -maxval/2 to maxval/2 for chrominance).

Note that the macros do it by fbating point multiplication. If you are computing these values over an entire image, it may be significantly faster to do it with multiplication tables instead. Compute all the possible products once up front, then for each pixel, just look up the products in the tables.

INITIALIZATION

All PPM programs must call **ppm_init()** just after invocation, before they process their arguments.

MEMORY MANAGEMENT

ppm_allocarray() allocates an array of pixels.

ppm_allocrow() allocates a row of the given number of pixels.

ppm_freearray() frees the array allocated with **ppm_allocarray()** containing the given number of rows.

ppm_freerow() frees a row of pixelss allocated with **ppm_allocrow()**.

READING FILES

If a function in this section is called on a PBM or PGM format file, it translates the PBM or PGM file into a PPM file on the fly and functions as if it were called on the equivalent PPM file. The *format* value returned by **ppm_readppminit()** is, however, not translated. It represents the actual format of the PBM or PGM file.

ppm_readppminit() reads the header of a PPM file, returning all the information from the header and leaving the file positioned just after the header.

ppm_readppmrow() reads a row of pixels into the *pixelrow* array. *format*, *cols*, and *maxval* are the values returned by **ppm_readppminit()**.

ppm_readppm() reads an entire PPM image into memory, returning the allocated array as its return value and returning the information from the header as *rows*, *cols*, and *maxval*. This function combines **ppm_readppminit()**, **ppm_allocarray()**, and **ppm_readppmrow()**.

WRITING FILES

ppm_writeppminit() writes the header for a PPM file and leaves it positioned just after the header.

forceplain is a logical value that tells **ppm_writeppminit()** to write a header for a plain PPM format file, as opposed to a raw PPM format file.

ppm_writeppmrow() writes the row *pixelrow* to a PPM file. For meaningful results, *cols*, *maxval*, and *forceplain* must be the same as was used with **ppm_writeppminit()**.

ppm_writeppm() write the header and all data for a PPM image. This function combines **ppm_writeppminit()** and **ppm_writeppmrow()**.

MISCELLANEOUS

ppm_nextimage() positions a PPM input file to the next image in it (so that a subsequent **ppm_readppminit()** reads its header).

ppm_nextimage() is analogous to **pbm_nextimage()**, but works on PPM, PGM, and PBM files.

ppm_check() checks for the common file integrity error where the file is the wrong size to contain all the image data.

ppm_check() is analogous to **pbm_check()**, but works on PPM, PGM, and PBM files.

COLOR NAMES

ppm_parsecolor() Interprets a color specification and returns a pixel of the color that it indicates. The color specification is ASCII text, in one of three formats: 1) a name, as defined in the system's X11-style color names file (e.g. **rgb.txt**). 2) an X11-style hexadecimal triple: #rgb, #rrggbb, #rrrgggbbb, or #rrrrggggbbbb. 3) A triplet of decimal floating point numbers from 0.0 to 1.0, representing red, green, and blue intensities respectively, separated by commas. E.g. "1.0, 0.5, .25".

If the color specification does not conform to any of these formats, including the case that it is a name, but is not in the rgb.txt database, **ppm_parsecolor()** exits the program via **pm_error()**.

ppm_colorname() Returns a string that describes the color of the given pixel. If an X11-style color names file (e.g. **rgb.txt**) is available and the color appears in it, **ppm_colorname()** returns the name of the color from the file. If the color does not appear in a X11-style color file and *hexok* is true, **ppm_colorname()** returns a hexadecimal color specification triple (#rrggbb). If a X11-style color file is available but the color does not appear in it and *hexok* is false, **ppm_colorname()** returns the name of the closest matching color in the color file. Finally, if their is no X11-style color file available and *hexok* is false, **ppm_colorname()** fails and exits the program with an error message.

The string returned is in static libppm library storage which is overwritten by every call to **ppm_colorname()**.

COLOR INDEXING

Sometimes in processing images, you want to associate a value with a particular color. Most often, that's because you're generating a color mapped graphics format. In a color mapped graphics format, the raster contains small numbers, and the file contains a color map that tells what color each of those small numbers refers to. If your image has only 256 colors, but each color takes 24 bits to describe, this can make your output file much smaller than a straightforward RGB raster would.

So, continuing the above example, say you have a **pixel** value for chartreuse and in your output file and you are going to represent chartreuse by the number 12. You need a data structure that allows your program quickly to find out that the number for a chartreuse **pixel** is 12. Netpbm's color indexing data types and functions give you that.

colorhash_table is a C data type that associates an integer with each of an arbitrary number of colors. It is a hash table, so it uses far less space than an array indexed by the color's RGB values would.

The problem with a **colorhash_table** is that you can only look things up in it. You can't find out what colors are in it. So Netpbm has another data type for representing the same information, the poorly but historically named **colorhist_vector**. A **colorhist_vector** is just an array. Each entry represents a color and contains the color's value (as a **pixel**) and the integer value associated with it. The entries are filled in starting with subscript 0 and going consecutively up for the number of colors in the histogram.

(The reason the name is poor is because a color histogram is only one of many things that could be represented by it).

colorhash_table ppm_alloccolorhash()

This creates a **colorhash_table** using dynamically allocated storage. There are no colors in it. If there is not enough storage, it exits the program with an error message.

void ppm_freecolorhash()

This destroys a **ppm_freecolorhash** and frees all the storage associated with it.

int ppm_addtocolorhash(colorhash_table cht, const pixel * const colorP, const int value)

This adds the specified color to the specified **colorhash_table** and associates the specified value with it.

You must ensure that the color you are adding isn't already present in the **colorhash_table**.

There is no way to update an entry or delete an entry from a **colorhash_table**.

int ppm_lookupcolor(const colorhash_table cht, const pixel * const colorP)

This looks up the specified color in the specified **colorhash_table**. It returns the integer value associated with that color.

If the specified color is not in the hash table, the function returns -1. (So if you assign the value -1 to a color, the return value is ambiguous).

colorhist_vector ppm_colorhashtocolorhist(const colorhash_table cht, const int ncolors)

This converts a **colorhash_table** to a **colorhist_vector**. The return value is a new **colorhist_vector** which you must eventually free with **ppm_freecolorhist()**.

ncolors is the number of colors in **cht**. If it has more colors than that, **ppm_colorhashtocolorhist** does not create a **colorhist_vector** and returns NULL.

colorhash_table ppm_colorhisttocolorhash(const colorhist_vector chv, const int ncolors)

This poorly named function does *not* convert from a **colorhist_vector** to a **colorhash_table**.

It does create a **colorhash_table** based on a **colorhist_vector** input, but the integer value for a given color in the output is not the same as the integer value for that same color in the input. **ppm_colorhisttocolorhash()** ignores the integer values in the input. In the output, the integer value for a color is the index in the input **colorhist_vector** for that color.

You can easily create a color map for an image by running **ppm_computecolorhist()** over the image, then **ppm_colorhisttocolorhash()** over the result. Now you can use **ppm_lookupcolor()** to find a unique color index for any pixel in the input.

If the same color appears twice in the input, **ppm_colorhisttocolorhash()** exit the program with an error message.

ncolors is the number of colors in **chv**.

The return value is a new **colorhash_table** which you must eventually free with **ppm_freecolorhash()**.

COLOR HISTOGRAMS

The Netpbm libraries give you functions to examine a Netpbm image and determine what colors are in it and how many pixels of each color are in it. This information is known as a color histogram. Netpbm uses its **colorhash_table** data type to represent a color histogram.

colorhash_table ppm_computecolorhash(pixel ** const pixels, const int cols, const int rows, const int maxcolors, int* const colorsP)

This poorly but historically named function generates a **colorhash_table** whose value for each color is the number of pixels in a specified image that have that color. (I.e. a color histogram). As a bonus, it returns the number of colors in the image.

(It's poorly named because not all **colorhash_table**s are color histograms, but that's all it generates).

pixels, **cols**, and **rows** describe the input image.

maxcolors is the maximum number of colors you want processed. If there are more colors that that in the input image, **ppm_computecolorhash()** returns NULL as its return value and stops processing as soon as it discovers this. This makes it run faster and use less memory. One use for **maxcolors** is when you just want to find out whether or not the image has more than N colors and don't want to wait to generate a huge color table if so. If you don't want any limit on the number of colors, specify **maxcolors**=**0**.

ppm_computecolorhash() returns the actual number of colors in the image as ***colorsP**, but only if it is less than or equal to **maxcolors**.

colorhash_table ppm_computecolorhash2(FILE * const ifp, const int cols, const int rows, const pixval maxval, const int format, const int maxcolors, int* const colorsP)

This is the same as **ppm_computecolorhash()** except that instead of feeding it an array of pixels in storage, you give it an open file stream and it reads the image from the file. The file must be positioned after the header, at the raster. Upon return, the file is still open, but its position is undefined.

maxval and **format** are the values for the image (i.e. information from the file's header).

colorhist_vector ppm_computecolorhist(pixel ** pixels, int cols, int rows, int maxcolors, int * colorsP)

This is like **ppm_computecolorhash()** except that it creates a **colorhist_vector** instead of a **colorhash_table**.

If you supply a nonzero **maxcolors** argument, that is the maximum number of colors you expect to find in the input image. If there are more colors than you say in the image, **ppm_computecolorhist()** returns a null pointer as its return value and nothing meaningful as ***colorsP**.

If not, the function returns the new **colorhist_vector** as its return value and the actual number of colors in the image as ***colorsP**. The returned array has space allocated for the specified number of colors regardless of how many actually exist. The extra space is at the high end of the array and is available for your use in expanding the **colorhist_vector**.

If you specify **maxcolors**=**0**, there is no limit on the number of colors returned and the return array has space for 5 extra colors at the high end for your use in expanding the **colorhist_vector**.

colorhist_vector ppm_computecolorhist2(FILE * ifp,
int cols, int rows, int maxcolors, pixval maxval, int format,
int * colorsP)

This is the same as **ppm_computecolorhist()** except that instead of feeding it an array of pixels in storage, you give it an open file stream and it reads the image from the file. The file must be positioned after the header, at the raster. Upon return, the file is still open, but its position is undefined.

SEE ALSO

pbm(5), **pgm**(5), **libpbm**(3)

AUTHOR

Copyright (C) 1989, 1991 by Tony Hansen and Jef Poskanzer.

NAME

pam - portable arbitrary map file format

DESCRIPTION

The PAM image format is a lowest common denominator 2 dimensional map format.

It is designed to be used for any of myriad kinds of graphics, but can theoretically be used for any kind of data that is arranged as a two dimensional rectangular array. Actually, from another perspective it can be seen as a format for data arranged as a three dimensional array.

This format does not define the meaning of the data at any particular point in the array. It could be red, green, and blue light intensities such that the array represents a visual image, or it could be the same red, green, and blue components plus a transparency component, or it could contain annual rainfalls for places on the surface of the Earth. Any process that uses the PAM format must further define the format to specify the meanings of the data.

A PAM image describes a two dimensional grid of tuples. The tuples are arranged in rows and columns. The width of the image is the number of columns. The height of the image is the number of rows. All rows are the same width and all columns are the same height. The tuples may have any degree, but all tuples have the same degree. The degree of the tuples is called the depth of the image. Each member of a tuple is called a sample. A sample is an unsigned integer which represents a locus along a scale which starts at zero and ends at a certain maximum value greater than zero called the maxval. The maxval is the same for every sample in the image. The two dimensional array of all the Nth samples of each tuple is called the Nth plane or Nth channel of the image.

Though the format does not assign any meaning to the tuple values, it does include an optional string that describes that meaning. The contents of this string, called the tuple type, are arbitrary from the point of view of the PAM format, but users of the format may assign meaning to it by convention so they can identify their particular implementations of the PAM format.

The Layout

A PAM file consists of a sequence of one or more PAM images. There are no data, delimiters, or padding before, after, or between images.

Each PAM image consists of a header followed immediately by a raster.

Here is an example header:

```
P7
WIDTH 227
HEIGHT 149
DEPTH 3
MAXVAL 255
TUPLETYPE RGB
ENDHDR
```

The header begins with the ASCII characters "P7" followed by newline. This is the magic number.

The header continues with an arbitrary number of lines of ASCII text. Each line ends with and is delimited by a newline character.

Each header line consists of zero or more whitespace-delimited tokens or begins with "#". If it begins with "#" it is a comment and the rest of this specification does not apply to it.

A header line which has zero tokens is valid but has no meaning.

The type of header line is identified by its first token, which is 8 characters or less:

ENDHDR
This is the last line in the header. The header must contain exactly one of these header lines.

HEIGHT
The second token is a decimal number representing the height of the image (number of rows). The header must contain exactly one of these header lines.

WIDTH
The second token is a decimal number representing the width of the image (number of columns). The header must contain exactly one of these header lines.

DEPTH
The second token is a decimal number representing the depth of the image (number of planes or channels). The header must contain exactly one of these header lines.

MAXVAL
The second token is a decimal number representing the maxval of the image. The header must contain exactly one of these header lines.

TUPLTYPE
The header may contain any number of these header lines, including zero. The rest of the line is part of the tuple type. The rest of the line is not tokenized, but the tuple type does not include any white space immediately following **TUPLTYPE** or at the very end of the line. It does not include a newline. If there are multiple **TUPLTYPE** header lines, the tuple type is the concatenation of the values from each of them, separated by a single blank, in the order in which they appear in the header. If there are no **TUPLETYPE** header lines the tuple type is the null string.

The raster consists of each row of the image, in order from top to bottom, consecutive with no delimiter of any kind between, before, or after, rows.

Each row consists of every tuple in the row, in order from left to right, consecutive with no delimiter of any kind between, before, or after, tuples.

Each tuple consists of every sample in the tuple, in order, consecutive with no delimiter of any kind between, before, or after, samples.

Each sample consists of an unsigned integer in pure binary format, with the most signifi cant byte fi rst. The number of bytes is the minimum number of bytes required to represent the maxval of the image.

PAM Used For PNM (PBM, PGM, or PPM) Images

A common use of PAM images is to represent the older and more concrete PBM, PGM, and PPM images.

A PBM image is conventionally represented as a PAM image of depth 1 with maxval 1 where the one sample in each tuple is 0 to represent a black pixel and 1 to represent a white one. The height, width, and raster bear the obvious relationship to those of the PBM image. The tuple type for PBM images represented as PAM images is conventionally "BLACKANDWHITE".

A PGM image is conventionally represented as a PAM image of depth 1. The maxval, height, width, and raster bear the obvious relationship to those of the PGM image. The tuple type for PGM images represented as PAM images is conventionally "GRAYSCALE".

A PPM image is conventionally represented as a PAM image of depth 3. The maxval, height, width, and raster bear the obvious relationship to those of the PPM image. The first plane represents red, the second blue, and the third green. The tuple type for PPM images represented as PAM images is conventionally "RGB".

The Confusing Universe of Netpbm Formats

It is easy to get confused about the relationship between the PAM format and PBM, PGM, PPM, and PNM. Here is a little enlightenment:

"PNM" is not really a format. It is a shorthand for the PBM, PGM, and PPM formats collectively. It is also the name of a group of library functions that can each handle all three of those formats.

"PAM" is in fact a fourth format. But it is so general that you can represent the same information in a PAM image as you can in a PBM, PGM, or PPM image. And in fact a program that is designed to read PBM, PGM, or PPM and does so with a recent version of the Netpbm library, will read an equivalent PAM image just fine and the program will never know the difference.

To confuse things more, there is a collection of library routines called the "pam" functions that read and write the PAM format, but also read and write the PBM, PGM, and PPM formats. They do this because the latter formats are much older and more popular, so this makes it convenient to write programs that use the newer PAM format.

SEE ALSO

pbm(5), **pgm**(5), **ppm**(5), **pnm**(5), **libpnm**(3).TH**pbm**5**05 March 2000**

NAME

pbm - portable bitmap file format

DESCRIPTION

The portable bitmap format is a lowest common denominator monochrome file format. It serves as the common language of a large family of bitmap conversion filters. Because the format pays no heed to efficiency, it is simple and general enough that one can easily develop programs to convert to and from just about any other graphics format, or to manipulate the image.

This is not a format that one would normally use to store a file or to transmit it to someone -- it's too expensive and not expressive enough for that. It's just an intermediary format. In it's purest use, it lives only in a pipe between two other programs.

The format definition is as follows.

A PBM file consists of a sequence of one or more PBM images. There are no data, delimiters, or padding before, after, or between images.

Each PBM image consists of the following:

- A "magic number" for identifying the file type. A pbm image's magic number is the two characters "P4".
- Whitespace (blanks, TABs, CRs, LFs).
- The width in pixels of the image, formatted as ASCII characters in decimal.
- Whitespace.
- The height in pixels of the image, again in ASCII decimal.
- Newline or other single whitespace character.
- A raster of Height rows, in order from top to bottom. Each row is Width bits, packed 8 to a byte, with don't care bits to fill out the last byte in the row. Each bit represents a pixel: 1 is black, 0 is white. The order of the pixels is left to right. The order of their storage within each file byte is most significant bit to least significant bit. The order of the file bytes is from the beginning of the file toward the end of the file.
- Characters from a "#" to the next end-of-line, before the width/height line, are comments and are ignored.

There is actually another version of the PBM format, even more more simplistic, more lavishly wasteful of space than PBM, called Plain PBM. Plain PBM actually came first, but even its inventor couldn't stand its recklessly squanderous use of resources after a while and switched to what we now know as the regular PBM format. But Plain PBM is so redundant -- so overstated -- that it's virtually impossible to break. You can send it through the most liberal mail system (which was the original purpose of the PBM format) and it will arrive still readable. You can flip a dozen random bits and easily piece back together the original image. And we hardly need to define the format here, because you can decode it by inspection.

The difference is:

- There is exactly one image in a file.
- The "magic number" is "P1" instead of "P4".
- Each pixel in the raster is represented by a byte containing ASCII '1' or '0', representing black and white respectively. There are no fill bits at the end of a row.
- White space in the raster section is ignored.
- You can put any junk you want after the raster, if it starts with a white space character.
- No line should be longer than 70 characters.

Here is an example of a small bitmap in the plain PBM format:

```
P1
# feep.pbm
24 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0
0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

You can generate the Plain PBM format from the regular PBM format (first image in the file only) with the **pnmtoplainpnm** program.

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a bitmap.

COMPATIBILITY

Before July 2000, there could be at most one image in a PBM file. As a result, most tools to process PBM files ignore (and don't read) any data after the first image.

SEE ALSO

libpbm(3),**pnm**(5),**pgm**(5),**ppm**(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pgm - portable graymap file format

DESCRIPTION

The PGM format is a lowest common denominator grayscale file format. It is designed to be extremely easy to learn and write programs for. (It's so simple that most people will simply reverse engineer it because it's easier than reading this specification).

A PGM image represents a grayscale graphic image. There are many psueudo-PGM formats in use where everything is as specified herein except for the meaning of individual pixel values. For most purposes, a PGM image can just be thought of an array of arbitrary integers, and all the programs in the world that think they're processing a grayscale image can easily be tricked into processing something else.

One official variant of PGM is the transparency mask. A transparency mask in Netpbm is represented by a PGM image, except that in place of pixel intensities, there are opaqueness values. See below.

The format definition is as follows.

A PGM file consists of a sequence of one or more PGM images. There are no data, delimiters, or padding before, after, or between images.

Each PGM image consists of the following:

- A "magic number" for identifying the file type. A pgm image's magic number is the two characters "P5".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- The maximum gray value (Maxval), again in ASCII decimal. Must be less than 65536.
- Newline or other single whitespace character.
- A raster of Width * Height gray values, proceeding through the image in normal English reading order. Each gray value is a number from 0 through Maxval, with 0 being black and Maxval being white. Each gray value is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.
- Each gray value is a number proportional to the intensity of the pixel, adjusted by the CIE Rec. 709 gamma transfer function. (That transfer function specifies a gamma number of 2.2 and has a linear section for small intensities). A value of zero is therefore black. A value of Maxval represents CIE D65 white and the most intense value in the image and any other image to which the image might be compared.
- Note that a common variation on the PGM format is to have the gray value be "linear," i.e. as specified above except without the gamma adjustment. **pnmgamma** takes such a PGM variant as input and produces a true PGM as output.
- In the transparency mask variation on PGM, the value represents opaqueness. It is proportional to the fraction of intensity of a pixel that would show in place of an underlying pixel, with the same gamma transfer function mentioned above applied. So what normally means white represents total opaqueness and what normally means black represents total transparency. In between, you would compute the intensity of a composite pixel of an "under" and "over" pixel as under * (1-(alpha/alpha_maxval)) + over * (alpha/alpha_maxval).<
- Characters from a "#" to the next end-of-line, before the maxval line, are comments and are ignored.

Note that you can use **pnmdepth** To convert between a the format with 1 byte per gray value and the one with 2 bytes per gray value.

There is actually another version of the PGM format that is fairly rare: "plain" PGM format. The format above, which generally considered the normal one, is known as the "raw" PGM format. See **pbm**(5) for some commentary on how plain and raw formats relate to one another.

The difference in the plain format is:

- There is exactly one image in a file.
- The magic number is P2 instead of P5.
- Each pixel in the raster is represented as an ASCII decimal number (of arbitrary size).
- Each pixel in the raster has white space before and after it. There must be at least one character of white space between any two pixels, but there is no maximum.
- No line should be longer than 70 characters.

Here is an example of a small graymap in this format:

```
P2
# feep.pgm
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a graymap.

COMPATIBILITY

Before April 2000, a raw format PGM file could not have a maxval greater than 255. Hence, it could not have more than one byte per sample. Old programs may depend on this.

Before July 2000, there could be at most one image in a PGM file. As a result, most tools to process PGM files ignore (and don't read) any data after the first image.

SEE ALSO

fitstopgm(1), fstopgm(1), hipstopgm(1), lispmtopgm(1), psidtopgm(1), rawtopgm(1), pgmbentley(1), pgmcrater(1), pgmedge(1), pgmenhance(1), pgmhist(1), pgmnorm(1), pgmoil(1), pgmramp(1), pgmtexture(1), pgmtofits(1), pgmtofs(1), pgmtolispm(1), pgmtopbm(1), pnm(5), pbm(5), ppm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

pnm - portable anymap file format

DESCRIPTION

The *pnm* programs operate on portable bitmaps, graymaps, and pixmaps, produced by the *pbm*, *pgm*, and *ppm* segments. There is no file format associated with *pnm* itself.

SEE ALSO

anytopnm(1), rasttopnm(1), tifftopnm(1), xwdtopnm(1), pnmtops(1), pnmtorast(1), pnmtotiff(1), pnmtoxwd(1), pnmarith(1), pnmcat(1), pnmconvol(1), pnmcrop(1), pnmcut(1), pnmdepth(1), pnmenlarge(1), pnmfile(1), pnmflip(1), pnmgamma(1), pnmindex(1), pnminvert(1), pnmmargin(1), pnmnoraw(1), pnmpaste(1), pnmrotate(1), pnmscale(1), pnmshear(1), pnmsmooth(1), pnmtile(1), ppm(5), pgm(5), pbm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.

NAME

ppm - portable pixmap file format

DESCRIPTION

The portable pixmap format is a lowest common denominator color image file format.

It should be noted that this format is egregiously inefficient. It is highly redundant, while containing a lot of information that the human eye can't even discern. Furthermore, the format allows very little information about the image besides basic color, which means you may have to couple a file in this format with other independent information to get any decent use out of it. However, it is very easy to write and analyze programs to process this format, and that is the point.

It should also be noted that files often conform to this format in every respect except the precise semantics of the sample values. These files are useful because of the way PPM is used as an intermediary format. They are informally called PPM files, but to be absolutely precise, you should indicate the variation from true PPM. For example, "PPM using the red, green, and blue colors that the scanner in question uses."

The format definition is as follows.

A PPM file consists of a sequence of one or more PPM images. There are no data, delimiters, or padding before, after, or between images.

Each PPM image consists of the following:

- A "magic number" for identifying the file type. A ppm image's magic number is the two characters "P6".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- The maximum color value (Maxval), again in ASCII decimal. Must be less than 65536.
- Newline or other single whitespace character.
- A raster of Width * Height pixels, proceeding through the image in normal English reading order. Each pixel is a triplet of red, green, and blue samples, in that order. Each sample is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.
- In the raster, the sample values are "nonlinear." They are proportional to the intensity of the CIE Rec. 709 red, green, and blue in the pixel, adjusted by the CIE Rec. 709 gamma transfer function. (That transfer function specifies a gamma number of 2.2 and has a linear section for small intensities). A value of Maxval for all three samples represents CIE D65 white and the most intense color in the color universe of which the image is part (the color universe is all the colors in all images to which this image might be compared).
- Note that a common variation on the PPM format is to have the sample values be "linear," i.e. as specified above except without the gamma adjustment. **pnmgamma** takes such a PPM variant as input and produces a true PPM as output.

- Characters from a "#" to the next end-of-line, before the maxval line, are comments and are ignored.

Note that you can use **pnmdepth** to convert between a the format with 1 byte per sample and the one with 2 bytes per sample.

There is actually another version of the PPM format that is fairly rare: "plain" PPM format. The format above, which generally considered the normal one, is known as the "raw" PPM format. See **pbm**(5) for some commentary on how plain and raw formats relate to one another.

The difference in the plain format is:

- There is exactly one image in a file.
- The magic number is P3 instead of P6.
- Each sample in the raster is represented as an ASCII decimal number (of arbitrary size).
- Each sample in the raster has white space before and after it. There must be at least one character of white space between any two samples, but there is no maximum. There is no particular separation of one pixel from another -- just the required separation between the blue sample of one pixel from the red sample of the next pixel.
- No line should be longer than 70 characters.

Here is an example of a small pixmap in this format:

```
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a pixmap.

COMPATIBILITY

Before April 2000, a raw format PPM file could not have a maxval greater than 255. Hence, it could not have more than one byte per sample. Old programs may depend on this.

Before July 2000, there could be at most one image in a PPM file. As a result, most tools to process PPM files ignore (and don't read) any data after the first image.

SEE ALSO

giftopnm(1), gouldtoppm(1), ilbmtoppm(1), imgtoppm(1), mtvtoppm(1), pcxtoppm(1), pgmtoppm(1), pi1toppm(1), picttoppm(1), pjtoppm(1), qrttoppm(1), rawtoppm(1), rgb3toppm(1), sldtoppm(1), spctoppm(1), sputoppm(1), tgatoppm(1), ximtoppm(1), xpmtoppm(1), yuvtoppm(1), ppmtoacad(1), ppmtogif(1), ppmtoicr(1), ppmtoilbm(1), ppmtopcx(1), ppmtopgm(1), ppmtopi1(1), ppmtopict(1), ppmtopj(1), ppmtopuzz(1), ppmtorgb3(1), ppmtosixel(1), ppmtotga(1), ppmtouil(1), ppmtoxpm(1), ppmtoyuv(1), ppmdither(1), ppmforge(1), ppmhist(1), ppmmake(1), ppmpat(1), ppmquant(1), ppmquantall(1), ppmrelief(1), pnm(5), pgm(5), pbm(5)

AUTHOR

Copyright (C) 1989, 1991 by Jef Poskanzer.